Deep Learning about Deep Learning projects

A complete analysis of open source deep learning projects

Megha Nagabhushan
School of Computing and Engineering
University of Missouri, Kansas City
mnbpc@mail.umkc.edu

Rashmi Tripathi
School of Computing and Engineering
University of Missouri, Kansas City
rtrh6@mail.umkc.edu

***Abstract*— There are a lot of open source deep learning projects available on the internet. These existing projects can be used to develop new innovative projects. The challenge is to understand the functionalities implemented in these projects. Also, these projects use many deep learning models. By knowing the functionalities and the deep learning technologies used, the developers can decide if they want to use the part of the logic in their project. In this project, we are building a model that can analyze a source code and provide results containing the functional semantics of the code as well as their dependencies. The model also segregates the code into clusters based on their functionalities**.

Keywords— open source; deep learning; semantics;

I. INTRODUCTION

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data. Deep learning implements principles of neural network like ability to express, Efficiency and Learnability. Deep learning is capable of Natural language generation, automatic speech recognition, image recognition and is also extensively used in health care industry. It has a lot of potential to solve complex problems and therefore it is in high demand.

Google had two deep-learning projects underway in 2012. Today it is pursuing more than 1,000 deep projects in all its major product sectors, including search, Android, Gmail, translation, maps, YouTube, and self-driving cars. [Figure-1]

Today, there are a lot of open source deep learning projects available on web based repositories. GitHub, one of the popular online repository hosting service has 867 deep leaning projects available. Developers reuse the existing code from these open source projects to develop their own new projects.

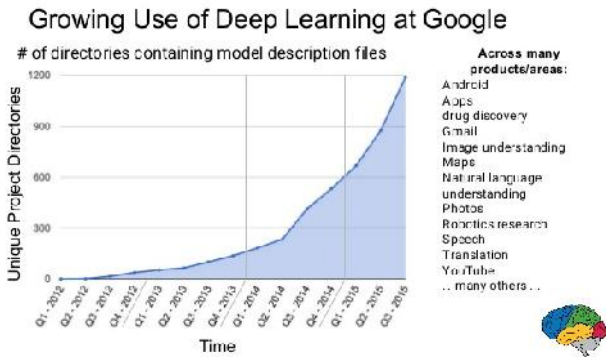


Figure-1 Deep Learning Curve at Google.

Understanding the network and logical flow of the code is a challenge. In our project, we are building an intelligent model that helps in internal learning and interpretation of the existing model. This intelligent model can be used to cascade the features of different projects and develop a new project.

II. RELATED WORK

Our model's working is similar to the way in which python programs are represented using **Pycallgraph**.

Pycallgraph: It is an open source library that creates call graph visualizations for Python applications. However our model scans the whole source code to show both static and dynamic call graphs.

Static Call Graph: It represents every possible run of the program. The exact static call graph is an undecidable problem, so static call graph algorithms are generally over approximations. That is, every call relationship that occurs is represented in the graph, and possibly also some call relationships that would never occur in actual runs of the program.

Dynamic Call Graph: A dynamic call graph is a record of an execution of the program, e.g., as output by a profiler. Thus, a dynamic call graph can be exact, but only describes one run of the program.

III. PROPOSED WORK

To develop a new model from the existing models, it is important to know the flow of logic and semantics of the code. In this project; we have decided to use deep learning technologies to design a model which is capable of analyzing the given code

and its behavior from two perspectives.

First one is static call graph which represents how the deep learning is done at every stage w.r.t source code. It is generalization of the source code with entry points and exit points. Here all the combinations of paths from entry point to exit point are shown. In contrast to second one i.e. dynamic call graph it will show one path from entry point and exit point. This path is decided by the input program given by user. Static and dynamic call graphs are best in their own way in terms of exposure they can value add to application users.

Furthermore for any organization, it will be interesting to adapt to a system which can train people on open source projects quickly. The combination of virtual reality world and call graph scan be proven best for the user's understanding. Users should have an application where they can observe these call graphs for any open source project in virtual reality world. With the motion of their hand like may be a swipe action, the application will reflect all the call graphs for given user program to human eyes. User should be able to see the various entry points, middle ones and the exit ones. Based upon the performance and the shortest path between an entry and an exit point User can easily learn and start making decision on what can be the best program for any particular goal.

A. *Model Architecture*

The architecture diagram shown in Figure-2 explains how our intelligent model works. The model takes a deep learning model source code as input. The source code is analyzed to determine functionality in terms of package structure, module, class structure, and module structure. The code with similar functionality and structure is grouped together into a cluster to reduce complexity. The next step will provide an interesting visualization to these clusters in form of static and dynamic call graphs.

B. *Uniqueness of the Model*

Deep learning projects use artificial neural network models that are employed in order to solve a wide spectrum of problems in optimization. Some of the commonly used models are multilayer perceptron (MLP) network and its variations (the time-lagged feedforward network (TLFN)), the generalized radial basis function (RBF) network, the recurrent neural network (RNN) and its variations (the time delay recurrent neural network (TDRNN)), and the counter propagation fuzzy-neural network (CFNN).

These models work differently. CNN Model will learn to recognize patterns called components (e.g., methods of a class) and learn to combine these components to recognize larger structures (e.g., classes). While CNN Model looks for the same patterns on all the sub fields, RNN Model feeds the hidden layer of the previous step as an additional input to the next step (e.g., recognizes the sequence of the method calls).

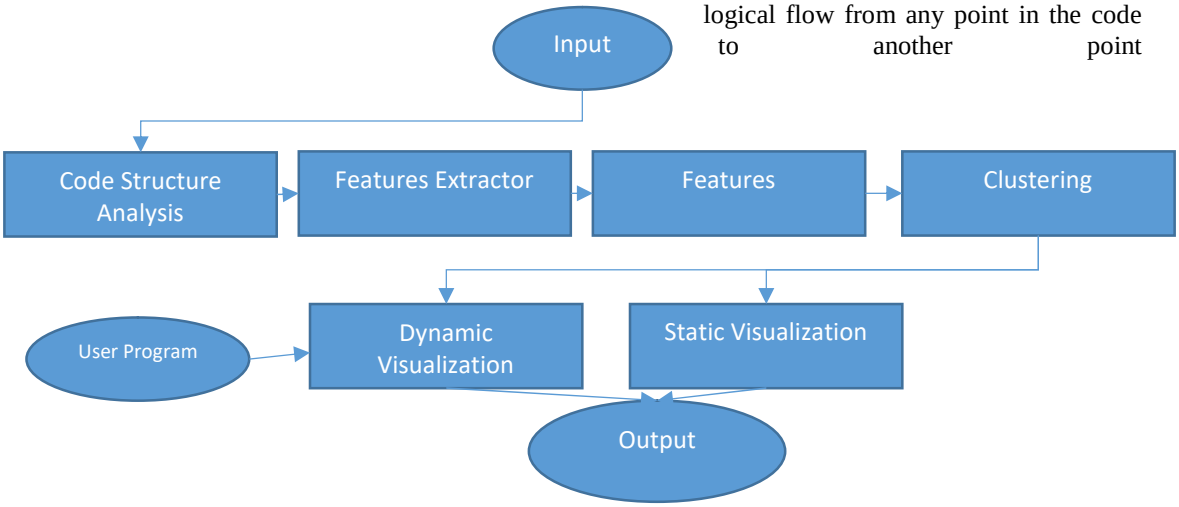
It is very important to know which of these models will be suitable for our model. The project also focusses on providing the information of the type of artificial neural network model used in a particular open source project.

IV. Implementation and Evaluation

A. *Scikit*

For our experiment, we are using Scikit-learn project. Scikit-learn is a free software machine learning library for the Python programming language. Scikit has a huge code base with 24 modules. Feature-selection is one module of Python with 15 python files and a total of 83 methods. We are using pycallgraph library to generate pictorial representation call graph [Figure -3]. Pycallgraph is a Python module that creates call graph visualizations for Python applications.

In our project, we will build a model that will enhance the understanding of the logical flow and semantics of the input code through innovative Interface. The model will also help in knowing the logical flow from any point in the code to another point



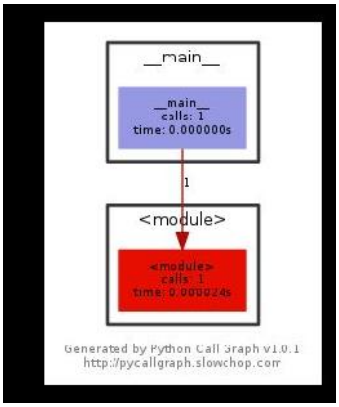


Figure 3: Pycall Graph representation

The analyzation of Scikit project shows a total of 39 moules and 349 classes.

B. *Tensorflow*

TensorFlow is open source project created by Google Brains Team for machine learning purposes providing a Python API and C++ API. It is used for machine learning in depth. The experiment is executed in many iterations.

In first iteration the system will generate static call graph. A crawler is created to crawl through each python module to associate the methods with respective packages and classes. There are total number of 1356 API available and their respective packages are known. Counts are shown below as per experiment:

Structure	Total
Packages	319
Modules (Number of python files)	268
Classes	Still in progress
Important API's	1356

Furthermore clustering is needed on them to associate them with type of it like entry points, intermediate points and exit points. As part of this, we learned that the module structure of TensorFlow have following main components:

util: utility functions.

select: various selection methods of TensorFlow tensors and operations.

match: TensorFlow graph matching. Think of this as regular expressions for graphs (but not quite yet).

reroute: various ways of rerouting tensors to different consuming ops like swap or reroute_a2b.

subgraph: the Subgraph View class, which enables subgraph manipulations in a TensorFlow tf.Graph.

edit: various editing functions operating on subgraphs like detach, connect or bypass.

transform: the Transformer class, which enables transforming (or simply copying) a subgraph into another one.

V. Discussion and Limitation

Our model is very helps to analyze a given deep learning project and come up with results which not only provides the logical and semantic flow of the project but also provides a report which contains information regarding the type of deep learning model and algorithms used in that project.

Our model is very suitable when the developer is looking for cascading unique features of different deep learning projects.

The limitation of our model is that it can be used to analyze only deep learning projects.

Conclusion

Deep Learning the deep learning projects would give useful information about an existing deep learning project which can be used to build robust and useful deep learning projects.

References

[1] Herbsleb, James, Christian Kästner, and Christopher Bogart. "Intelligently Transparent Software Ecosystems." IEEE Software 33.1 (2016): 89-96.

[2] L. Dabbish et al., "Leveraging Transparency," IEEE Software, vol. 30, no. 1, 2013, pp. 37–43.

[3] K.T. Stolee, S. Elbaum, and D. Dobos, "Solving the Search for Source Code," ACM Trans. Software Eng. and Methodology, vol. 23, no. 3, 2014, article 26.

[4] L. Martie, T.D. LaToza, and A. van der Hoek, "CodeExchange: Supporting Reformulation of Code Queries in Context," to be published in Proc. 30th Int'l Conf. Automated Software Eng., 2015.

[5] H. Cleve and A. Zeller, "Locating Causes of Program Failures," Proc. 27th Int'l Conf. Software Eng. (ICSE 05), 2005, pp. 342–351.

[6] R.E. Kraut and P. Resnick, eds., Building Successful Online Communities: Evidence-Based Social Design, MIT Press, 2011.

[7] K. Crowston et al., "Free/Libre OpenSource Software Development: What We Know and What We Do Not Know," ACM Computing Surveys, vol. 44, no. 2, 2012, article 7.

[8] E.S. Bernstein, "The Transparency Paradox: A Role for Privacy in Organizational Learning and Operational Control," Administrative Science Q., vol. 57, no. 2, 2012, pp. 181–216.

[9] L. Dabbish et al., "Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository," Proc. ACM 2012 Conf. Computer Supported Cooperative Work, 2012, pp. 1277–1286.

[10] J. Tsay, L. Dabbish, and J. Herbsleb, "In uence of Social and Technical Factors for Evaluating Contribution in GitHub," Proc. 36th Int'l Conf. Software Eng. (ICSE 14), 2014, pp. 356–366.

[11] R. Holmes and R.J. Walker, “Customized Awareness: Recommending Relevant External Change Events,” Proc. 32nd ACM/IEEE ACM Int’l Conf. Software Eng. (ICSE 10), 2010, pp. 465–474.
[12] R. Padhye, S. Mani, and V.S. Sinha, “NeedFeed: Taming Change Noti- cations by Modeling Code Relevance,” Proc. 29th ACM/IEEE Int’l Conf. Automated Software Eng. (ASE 14), 2014, pp. 665–676. 14. G.A. Hall and J.C. Munson, “Software Evolution: Code Delta and Code Churn,” J. Systems and Software, vol. 54, no. 2, 2000, pp. 111–118.
[13] M. Zhou and A. Mockus, “Developer Fluency: Achieving True Mastery in Software Projects,” Proc. 18th ACM SIGSOFT Int’l Symp. Foundations of Software Eng. (FSE 10), 2010, pp. 137–146.
[14] E.M. Rogers, Diffusion of Innovations, 4th ed., 1995, Free Press. Selected CS articles and columns
[15] http://cloud.google.com/deep-learning
[16] https://github.com/gak/pycallgraph