Deep Transfer Learning for Source Code Modeling

Yasir Hussain[a,*], Zhiqiu Huang[a,b,c,*], Yu Zhou[a], Senzhang Wang[a]

[a] *College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing 211106, China*
[b] *Key Laboratory of Safety-Critical Software, NUAA, Ministry of Industry and Information Technology, Nanjing 211106, China*
[c] *Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210093, China*

Abstract

Recently deep learning-based approaches have shown great potential in the modeling of source code for various software engineering tasks. These techniques lack adequate generalization and resistance to acclimate the use of such models in a real-world software development environment. In this work, we propose a novel general framework that combines cloud computing and deep learning in an integrated development environment (IDE) to assist software developers in various source code modeling tasks. Additionally, we present *DeepVS*, an end-to-end deep learning-based source code suggestion tool that shows a real-world implementation of our proposed framework. *DeepVS* tool is capable of providing source code suggestions instantly in an IDE by using a pre-trained source code model. Moreover, the *DeepVS* tool is also capable of suggesting zero-day (unseen) code tokens. The *DeepVS* tool illustrates the effectiveness of the proposed framework and shows how it can help to link the gap between developers and researchers.

Keywords: Deep Neural Networks; Cloud Computing; Source Code Modeling; Software Language Model

1. Introduction

Recently, deep learning techniques have been widely applied for various source code modeling tasks such as source code suggestion [7, 9], error fixing [4, 8], method naming [1], etc. Source code suggestion and completion are vital features of an integrated development environment (IDE). Software developers rely mostly on such features while developing software. Most of the modern IDE's source code suggestion tools provide the next possible source code by leveraging the information already exists in the code editor of the IDE. Due to the limited historical data, the IDE may not be able to provide the correct suggestions or may possibly suggest irrelevant suggestions.

*Corresponding author
Email addresses: yaxirhuxxain@nuaa.edu.cn (Yasir Hussain), zqhuang@nuaa.edu.cn (Zhiqiu Huang), zhouyu@nuaa.edu.cn (Yu Zhou), szwang@nuaa.edu.cn (Senzhang Wang)

The deep learning [7, 9] based approaches have recently shown how they can significantly improve such tools by leveraging historical data (e.g. GitHub). However, there are various limitations and concerns:

- Most methods require a huge amount of computation power as well as time to train such models.
- Current techniques lack to illustrate, how they can assist software developers in a real-world software development environment?
- The existing works present no enlightenment, how these models can be integrated into an IDE.

In this work, our goal is to enable the usage of deep learning-based models directly in an IDE for various source code modeling tasks. Here, we choose the source code suggestion task as a motivation example. In this work, first, we introduce a novel framework that helps in using machine learning or deep learning-based models directly in an IDE. Secondly, we present *DeepVS* an end-to-end deep learning-based source code suggestion tool. The *DeepVS* tool leverages from a pre-trained deep neural language model train on over 13M code tokens for the task of source code suggestion directly in an IDE. The tool is capable of suggesting zero-day (unseen) code tokens.

This work makes the following unique contributions.

- To the best of our knowledge, this work is the first one to propose a general framework that enables the usage of machine learning and deep learning-based source code models directly in an IDE.
- We present *DeepVS* tool, end-to-end implementation of our proposed framework for source code suggestion task.
- The quantitative evaluation with a real-world dataset and qualitative analysis of *DeepVS* tool verifies that the projected method is accurate and effective in predicting code suggestions.

2. Approach

The overall high-level architecture of the proposed framework is presented in Fig. 1. The framework consists of three layers. The first layer consists of a machine learning or deep learning-based pre-trained model, which is used to estimate the likelihood scores for a given source code modeling task. The second layer is a cloud/local platform, which is used to serve pre-trained models locally or over a cloud platform for the purpose of likelihood estimation. The final layer is an IDE plugin interface to interacts with the cloud/local platform for a specific source code modeling task.

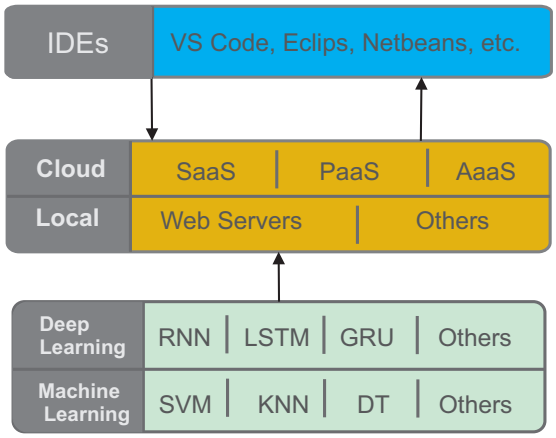


Figure 1: A general framework for usage of machine learning based source code models directly in an IDE.

the overall workflow of our *DeepVs* tool is presented in Fig. 2. On the first layer of *DeepVS*, we are using Gated Recurrent Unit (GRU) [3] based deep learning classifier in a bidirectional fashion which learns the source code context in both directions to help suggest the next source code token. On the second layer, we present a Cloud-based platform named *Deep Cloud*, which uses the pre-trained bidirectional Gated Recurrent Unit (BiGRU) neural net for the source code suggestion task. Finally, for demonstration purpose, we implement a plugin interface in *Visual Studio Code* IDE, which interacts with *Deep Cloud* for source code suggestion task in real-time.

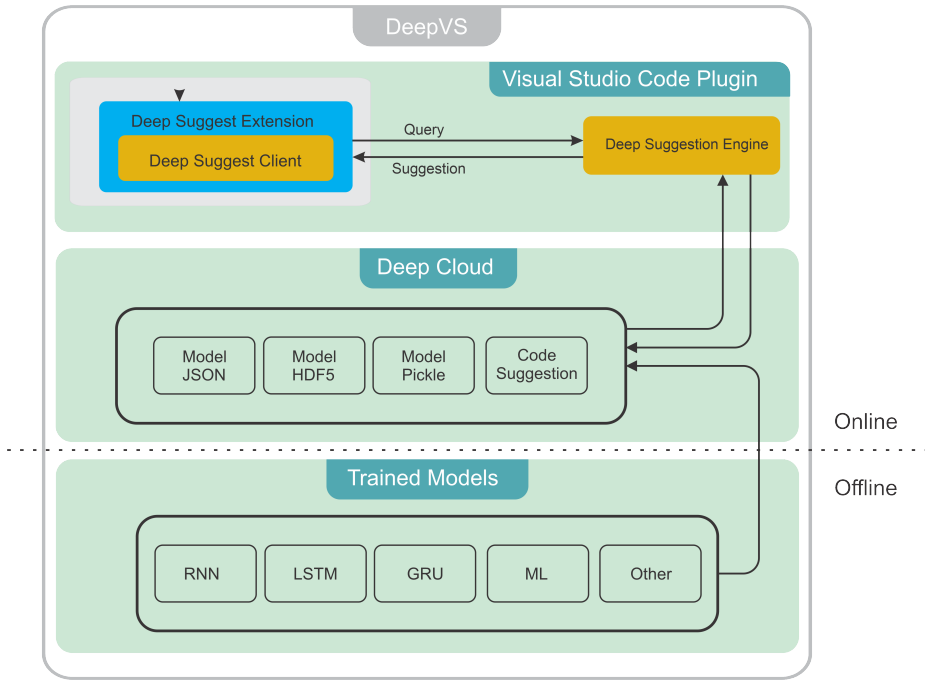


Figure 2: Architecture of our *DeepVS* tool which is a real world implementation of our proposed framework.

2.1. The Deep Cloud Layer

The *Deep Cloud* platform is used as a centralized deep learning model attendant to determine the likelihood scores for the next source code suggestion task. Here, we choose cloud platform [2] because of its centralized nature and scalability. Our *Deep Cloud* is a platform as a service-based cloud. *Deep Cloud* consists of a single core Intel CPU with 1GB Ram running ubuntu-18 x64-bit non-GUI edition. *Deep cloud* takes source code context as input and provides a *JSON* object which contains most likely next source code suggestions.

2.2. The plugin Layer

For the demonstration purpose, we implement the plugin interface in *Visual studio code* IDE. The *DeepVS* tool can be triggered by using shortcut key *CTRL+SPACE*. By triggering *DeepVS*, it takes the source code prior to the current cursor position as context and requests the *Deep Cloud* for most probable next source code tokens by leveraging the pre-trained BiGRU model.

3. Evaluation

The evaluation of this work is two-fold, quantitative and qualitative. In quantitative evaluation, we assess the classification outcomes to test the performance of the

proposed method and compare it with state-of-the-art methods. Metrics used to assess the quantitative efficiency of this work are top-k accuracy and mean reciprocal rank (MRR), which are mostly applied. For the qualitative evaluation of the work, we conduct an experimental study involving seven university students having software development experience. The experiment was performed with the help of a questionnaire. All the participants were demonstrated with the basic usage of the *DeepVS* plugin. The questionnaire given to the participants is shown in Table 2.

3.1. Dataset

For the training and testing of the proposed method, this work used the dataset anticipated in [5, 6]. The dataset comprises of ten java projects (*ant, cassandra, db40, jgit, poi, batik, antlr, itext, jts, maven*). The dataset consists of over 13M code tokens with large vocabulary of size 145,457 (singletons removed). Table 1 shows the statistics of our training, and testing sets in detail.

Table 1: Code database statistics

	Line of Code (LoC)	Total Code Tokens	Vocab Size
Training	1,684,158	12,086,347	
Testing	187,129	1,342,927	
Total	1,871,287	13,429,274	145,457

4. Results

4.1. Quantitative Results

In order to realistically evaluate the effectiveness of the proposed method, we compare the performance of our proposed method with other state-of-the-art methods. We choose Hindle el al. [5] and White et al. [9] work as baselines because they are related to our target task. The comprehensive results are given in Table 3. It can be witnessed that the proposed method proves noticeably better performance as compared to other baselines. Fig. 3 and Fig. 4 represents the accuracy and loss of training and validation accordingly.

Table 2: Questionnaire used for evaluation.

Questions	**Choices**
Q1: How easy it is to set up and use the DeepVS tool?	(1=Very Easy, 5=Very Difficult)
Q2: Does *DeepVS* tool provides suggestions in real-world coding environment?	(1=Strongly Agree, 5=Strongly Disagree)
Q3: Does *DeepVS* tool performs better than IDE's default source code suggestion engine?	(1=Strongly Agree, 5=Strongly Disagree)
Q4: I will recommend *DeepVS* to others.	(1=Strongly Agree, 5=Strongly Disagree)

Figure 3: Training and validation accuracy comparison.

Figure 4: Training and validation loss comparison.

4.2. Qualitative Results

As reported in Fig. 5, participants found the *DeepVS* tool is easy to set up and use. In Q2, all participants have agreed that *DeepVS* tool is capable of providing suggestions in a real-world development environment. In Q3, out of seven participants three voted strongly agree, three of them voted agree and one voted neutral. In Q4, most of the participants have agreed that they would recommend *DeepVS* to their peers. The illustrative examples of source code suggestion tasks are provided in Section 4.3.

Table 3: Accuracy and MRR scores comparison.

Model	Accuracy				MRR
	K-1	K-3	k-5	K-10	
N-gram [5]	48.47%	57.68%	59.87%	61.78%	0.535
RNN [9]	51.30%	67.67%	72.18%	76.21%	0.596
BiGRU	**73.77%**	**84.39%**	**86.97%**	**89.34%**	**0.792**

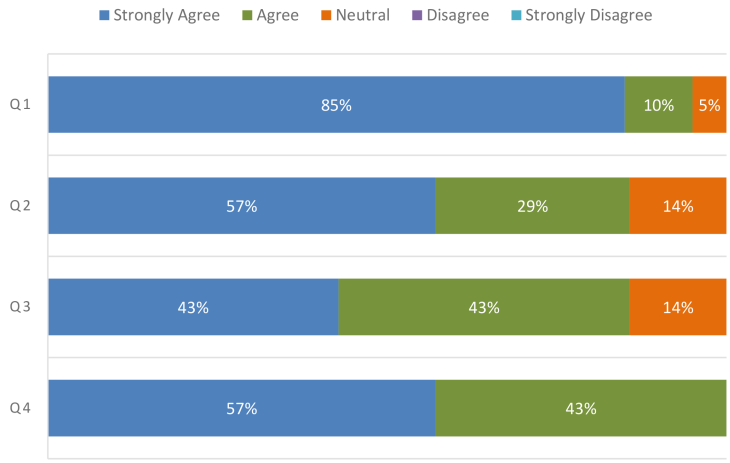


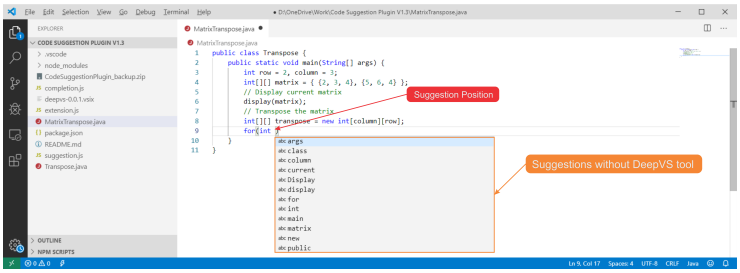
Figure 5: Questionnaire result.

4.3. DeepVS Tool Demo

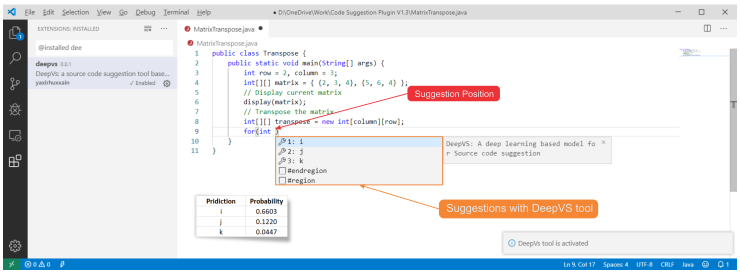
To show the effectiveness of our proposed framework for real-world usage, we present *DeepVS*, a deep learning-based source code suggestion tool for *Visual Studio Code* IDE. Consider the example presented in Fig. 6 were a software developer is writing a complex program to find the transpose of a matrix. Hereby triggering (line *9*) our *DeepVS* tool it suggest the most probable next source code tokens (Fig. 6a) *i,j and k* where the *Visual Studio Code's* default suggestion engine fails to provide any relevant suggestions (Fig. 6b), instead it torment the developer with irrelevant suggestions. Later on, (Fig. 7) at line *21* the software developer is writing a print statement. Here the most probable next source code token is *column* based on the given context. Hereby triggering our *DeepVS* tool it suggests the correct next source code token *column* (Fig. 7b) at its first index whereas the IDE's default source code suggestion tool rank the correct suggestion (Fig. 7a) on its third index.

4.3.1. Proposed Work's Major Benefits

The qualitative and quantitative evaluation verifies the feasibility and practicality of this work. We summarize the benefits of our proposed framework and the *DeepVS*

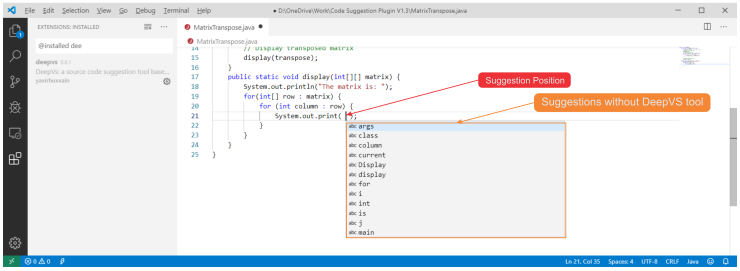


(a)

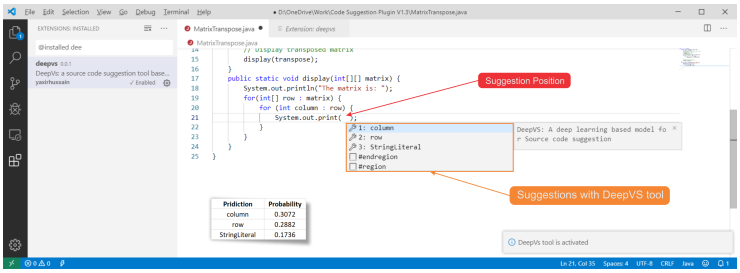


(b)

Figure 6: Source code suggestion example with and without the *DeepVs* tool along with their probabilities.



(a)



(b)

Figure 7: Another Source code suggestion example along with their probabilities.

tool as follows:

- The proposed framework enables the usage of machine/deep learning-based source code models directly in an IDE.
- An end-to-end deep learning-based source code suggestion tool named *DeepVS* shows a real-world implementation of the proposed framework, verifying its practicality.
- The *DeepVS* tool is trained on over 13M code tokens, thus capable of suggesting zero-day (unseen) code tokens by leveraging large scale historical codebase.
- The *DeepVS* tool ranks the suggestions efficiently by leveraging a deep learning-based source code model, without tormenting the developer with unnecessary suggestion.

5. Conclusion

In this work, we propose a novel general framework which helps in using machine or deep learning-based source code models for various software engineering tasks. Further, we presented *DeepVS*, an end-to-end deep learning-based source code suggestion tool, which leverages from a pre-trained deep learning model to help suggest the next source code token directly in an IDE in real-time. The *DeepVS* tool shows the effectiveness of the proposed framework and shows that it is of practical use.

Acknowledgments

This work was supported by the National Key R&D sponsored by Qing Lan Project of China [grant no. 2018YFB1003902].

References

[1] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. code2vec: Learning distributed representations of code. *Proceedings of the ACM on Programming Languages*, 3(POPL):40, 2019.

[2] Rami Bahsoon, Ivan Mistrík, Nour Ali, TS Mohan, and Nenad Medvidovic. The future of software engineering in and for the cloud. *Journal of Systems and Software*, 86(9):2221–2224, 2013.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[4] Rahul Gupta, Aditya Kanade, and Shirish Shevade. Deep reinforcement learning for programming language correction. *arXiv preprint arXiv:1801.10467*, 2018.

[5] Abram Hindle, Earl T Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. On the naturalness of software. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 837–847. IEEE, 2012.

[6] Anh Tuan Nguyen, Trong Duc Nguyen, Hung Dang Phan, and Tien N Nguyen. A deep neural network language model with contexts for source code. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 323–334. IEEE, 2018.

[7] Veselin Raychev, Martin Vechev, and Eran Yahav. Code completion with statistical language models. In *Acm Sigplan Notices*, volume 49, pages 419–428. ACM, 2014.

[8] Eddie Antonio Santos, Joshua Charles Campbell, Dhvani Patel, Abram Hindle, and José Nelson Amaral. Syntax and sensibility: Using language models to detect and correct syntax errors. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 311–322. IEEE, 2018.

[9] Martin White, Christopher Vendome, Mario Linares-Vásquez, and Denys Poshyvanyk. Toward deep learning software repositories. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, pages 334–345. IEEE Press, 2015.