# 論　文　の　要　旨

題　目　　A study of combining code representation techniques with deep learning for software defect prediction

（ソフトウェア欠陥予測のためのコード表現技術とディープラーニングの組み合わせに関する研究）

氏　名　　FANG DINGBANG

With the widespread application of software in various fields such as industry, finance and healthcare, it has had a profound impact on human life. As developers seek to meet customer demands for software functionality, the corresponding software becomes larger and more complex. Defect prediction plays an important role in development to ensure the quality and reliability of software. If this technology can effectively predict potential defects in software modules during the early stages of development, it will help group members allocate resources to address potential failures in the software, thus significantly resolving issues of time and cost associated with testing. However, these are only suitable for quantitative analysis of specific projects and also lack an understanding of program syntax and semantics.

Defect prediction relies on historical data from the software to build a model that predicts whether a defect exists in a module of the current project. Previous methods primarily utilized code metrics (static and process metrics) with statistical properties extracted from historical data as manual features, which were then fed into traditional machine-learning algorithms. Recently, some researchers have started to focus on capturing semantic features from code by deep learning methods to overcome the challenge that code metrics do not generalize well to other projects. Although these methods for defect prediction have been proposed, the performance of the implementations is not satisfactory. This is because these methods do not extract enough information from the software. To capture valid information from the code to improve the performance of the model, we need to focus on the following issues: (1) how to effectively exploit the complementary advantages of static metrics designed to measure the complex properties of code and high-level semantic features derived from code based on DL. (2) how to abstractly represent the syntax and semantics of code through semantic graphs. (3) how to leverage transfer learning to reduce the variability between project data for generalization purposes.

To address the above three key issues, this dissertation accordingly proposes three different solutions to improve the performance of defect prediction, as follows:

(1) As the complexity of software makes programs difficult to understand, appropriate representations that capture features from different levels of

abstraction in the code can effectively represent the information in the code. We introduce *Gated Homogeneous Fusion Network* (GHFNet) for defect prediction, combining high-level semantic feature extraction and weighted static feature extraction. Through the mechanism of homogeneous gating fusion, weights are adaptively assigned to the two types of features based on the correlation of these features to form fused features for defect prediction in the code. Experiment results show that the proposed algorithm learns multiple levels of features efficiently and outperforms the reference algorithm for defect prediction.

(2) Some researchers have leveraged deep learning (DL) to learn semantic features from the abstract syntax tree to identify potential defects. However, they directly serialize the nodes in the abstract syntax tree to form a sequence as DL input, ignoring the structural information of the tree. To solve the above problem, we propose a property-enhanced lightweight graph (PLG) based on an abstract syntax tree (AST) reflecting structural information in the source code. The PLG, which retains nodes related to program semantics, is more lightweight than the AST as well as enhances the strength of the connection between leaf nodes and the parent nodes of semantically stronger attributes. PLG is expected to significantly reduce the complexity of the original tree and enhance the attribute relationships between nodes. Moreover, based on PLG, we also develop a graph representation-based learning system to facilitate graph neural networks in defect prediction. The experiment results show that the proposed model has significant improvements for reference methods in several Java repositories.

(3) A deep learning system (DLS) developed based on one software project for defect prediction may well be applied to the related code on the same project but is usually difficult to be applied to new or unknown software projects. To address this problem, we propose a Transferable Graph Convolutional Neural Network (TGCNN) that can learn defects from the lightweight semantic graphs of code and transfer the learned knowledge from the source project to the target project. We discuss how the semantic graph is constructed from code; how the TGCNN can learn from the graph; and how the learned knowledge can be transferred to a new or unknown project. We also conduct a controlled experiment to evaluate our method. The result shows that despite some limitations, our method performs considerably better than existing methods.