**COMPILER DESIGN PROJECT**

# Topic : Compiler-based graph representations for deep learning models of code

In this paper natural language processing, novel methods in deep learning, like recurrent neural networks (RNNs) on sequences of words, have been very successful. In contrast to natural languages, programming languages usually have a well-defined structure. With this structure compilers can reason about programs, using graphs such as abstract syntax trees (ASTs) or control-data flow graphs (CDFGs). In this paper, we argue that we should use these graph structures instead of sequences for learning compiler optimization tasks. To this end, we use graph neural networks (GNNs) for learning predictive compiler tasks on two representations based on ASTs and CDFGs. Experiments show that this improves upon the state-of-the-art in the task of heterogeneous OpenCL mapping, while providing orders of magnitude faster inference times, crucial for compiler optimizations. When testing on benchmark suites not included for training, our AST-based model significantly outperforms the state-of-the-art by over 12 percentage points in terms of accuracy. It is the only one to perform clearly better than a random mapping. On the task of predicting thread coarsening factors, we show that all of the methods fail to produce an overall speedup.

VARUN KUMAR YASHIT KUMAR

2K19/IT/140 2K19/IT/149