
# NeuroSLP

## A Reinforcement Learning Heuristic for Superword-Level Parallelism

Kyle Astroth, Sam Gonzalez, Carson Hoffman, Tom (Xiangyu) Qin

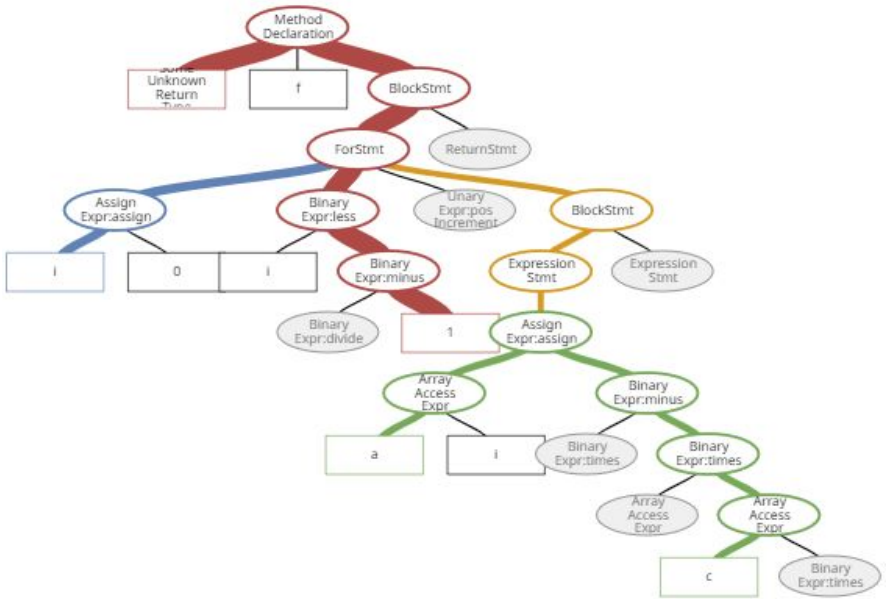
# SLP vectorization uses static parameters across programs unless explicitly changed

```
static cl::opt<int>
    SLPCostThreshold("slp-threshold", cl::init(0), cl::Hidden,
                     cl::desc("Only vectorize if you gain more than this "
                              "number "));

static cl::opt<int>
MaxVectorRegSizeOption("slp-max-reg-size", cl::init(128), cl::Hidden,
    cl::desc("Attempt to vectorize for this register size in bits"));

static cl::opt<unsigned>
MaxVFOption("slp-max-vf", cl::init(0), cl::Hidden,
    cl::desc("Maximum SLP vectorization factor (0=unlimited)"));
```

# Existing work on ML-based loop vectorization may be applicable to SLP parameter selection

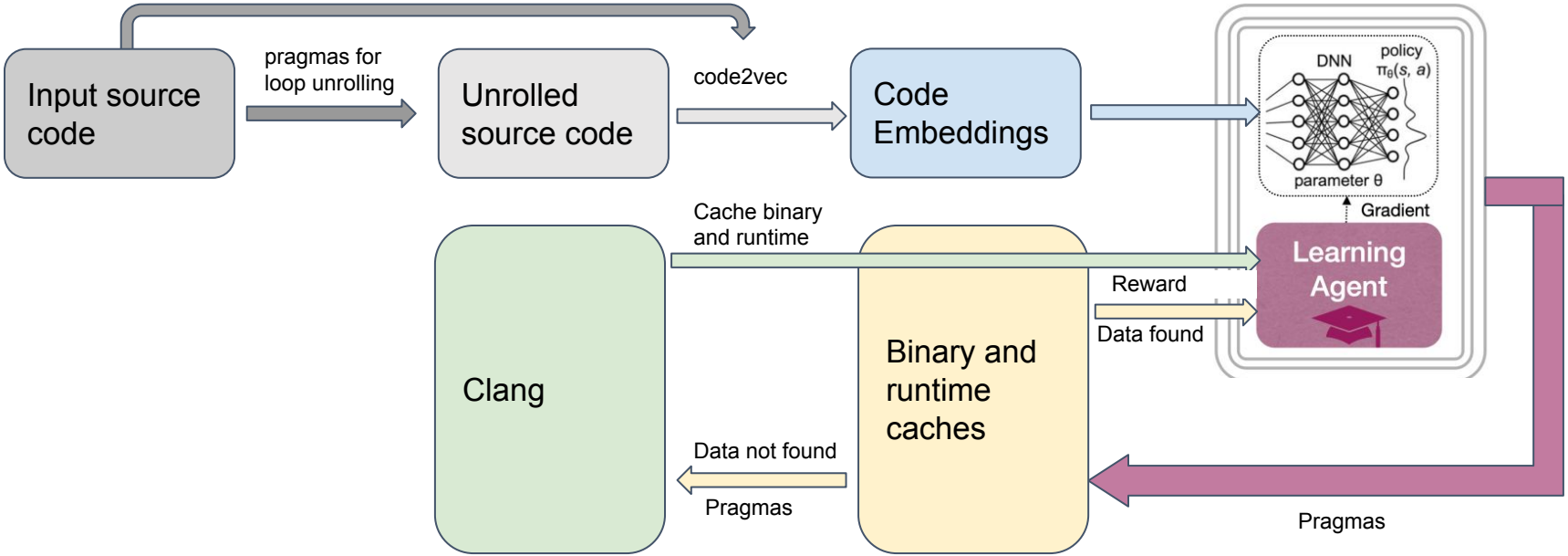


# Optimizations

- Hand picked two parameters out of 12 total SLP parameters
  - slp-max-reg-size and slp-threshold
  - Not over-complicate the network to learn something useful
  - Tried parameters on few input file - slp-upper-bounds, doesn't optimize

- Estimated 180+ hours to train large dataset (10,000 loops)
  - Experimenting with different parameters
- Bottleneck in training the model was compiling and running the code
- Caching of binary files and runtime for each pragma
  - Run binary file with each unique pragmas exactly once
- Noise Filtering - See if binary file is identical to binary file produced by -O3 optimization
  - Only run the same binary file once

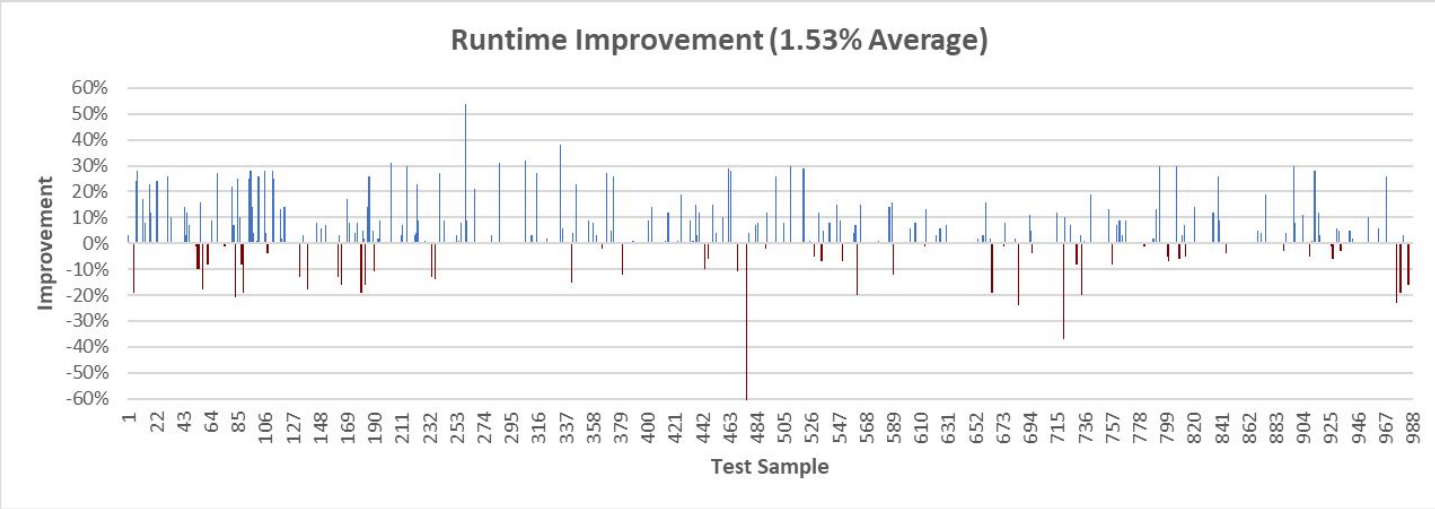
# Implementation



# Case Study / Demo

# Results

- Trained on 1,000 random loops from the Neuro Vectorizer dataset
  - 100,000 episodes with single thread under idle load (Approx. 56 hours)
- Evaluated on a different 1,000 random loops
- Average improvement of 1.53% (6.42% for modified binarys)



# Current Limitations

```
1  void foo(int a1, int a2, int b1, int b2, int *A) {
2    A[0] = a1*(a1 + b1);
3    A[1] = a2*(a2 + b2);
4    A[2] = a1*(a1 + b1);
5    A[3] = a2*(a2 + b2);
6  }
```

*Example of a function that would benefit from SLP vectorization*

- Training is slow
- Noisy CPU runtime metrics
- LLVM only allows control of SLP parameters for entire program, not at a basic block level
- Existing training set is not representative of most SLP workloads

# Concluding Remarks

- VF & IF parameters are more significant for decreasing run time with vectorization, however SLP can significantly improve runtime for niche applications.
- Better approach: implement NeuroSLP in tandem with NeuroVectorizer
- Other possible future improvements:
  - Replace Code2Vec embedding with LLVM AST
  - Separate training data for NeuroSLP that is representative of code that would benefit from SLP vectorization

# Questions?