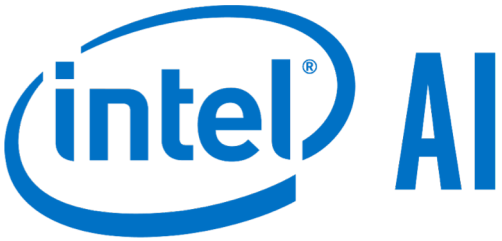
NeuroVectorizer: End-to-End Vectorization with Deep Reinforcement Learning

Ameer Haj-Ali, Nesreen K. Ahmed, Ted Willke,
Yakun Sophia Shao, Krste Asanovic, and Ion Stoica

University of California, Berkeley

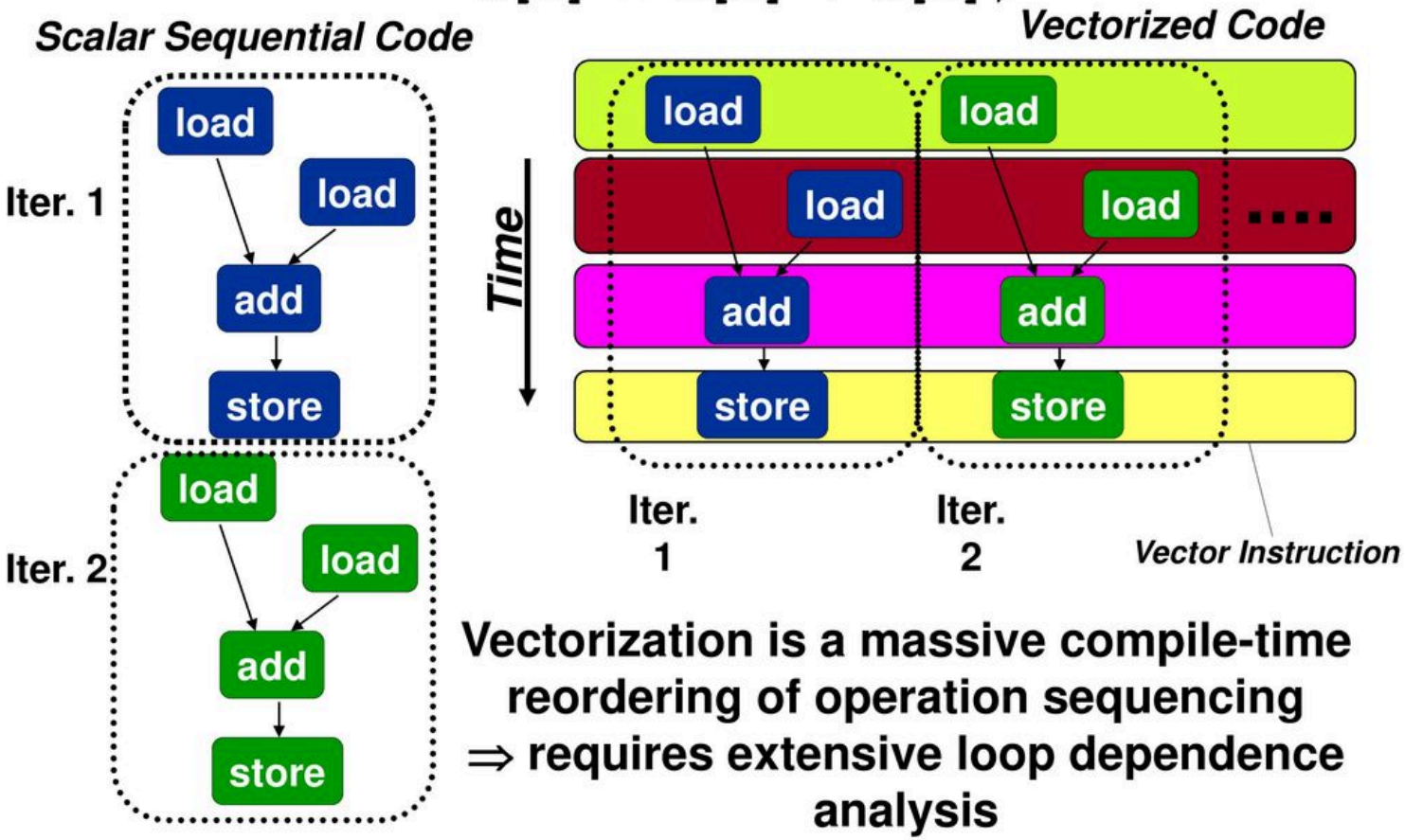
Intel Labs

Toward automatic and intelligent optimization of off-the-shelf code ...



How to Determine the Optimal Vector Size and Stride for Loops?

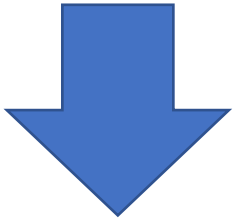
```
for (i=0; i < N; i++)
    C[i] = A[i] + B[i];
```



Vectorization is a massive compile-time reordering of operation sequencing ⇒ requires extensive loop dependence analysis

Vectorization and Interleaving Factors Example

```
for(int i =0; i<n; i++){
        sum += vec[i]*vec[i];
}
```

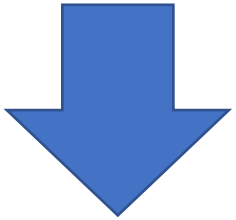


-O3

```
#pragma clang loop vectorize_width(4) interleave_count(2)
for(int i =0; i<n; i++){
        sum += vec[i]*vec[i];
}
```

Vectorization and Interleaving Factors Example

```
for(int i =0; i<n; i++){
        sum += vec[i]*vec[i];
}
```

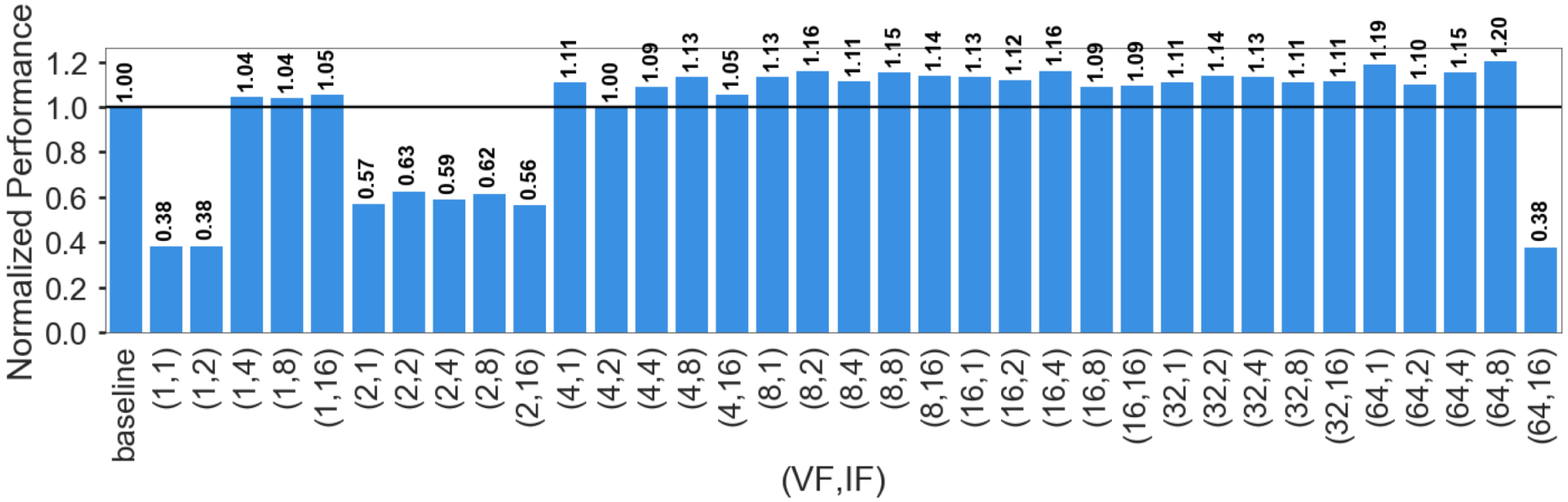


Optimal (brute force search)

```
#pragma clang loop vectorize_width(8) interleave_count(4)
for(int i =0; i<n; i++){
        sum += vec[i]*vec[i];
}
```

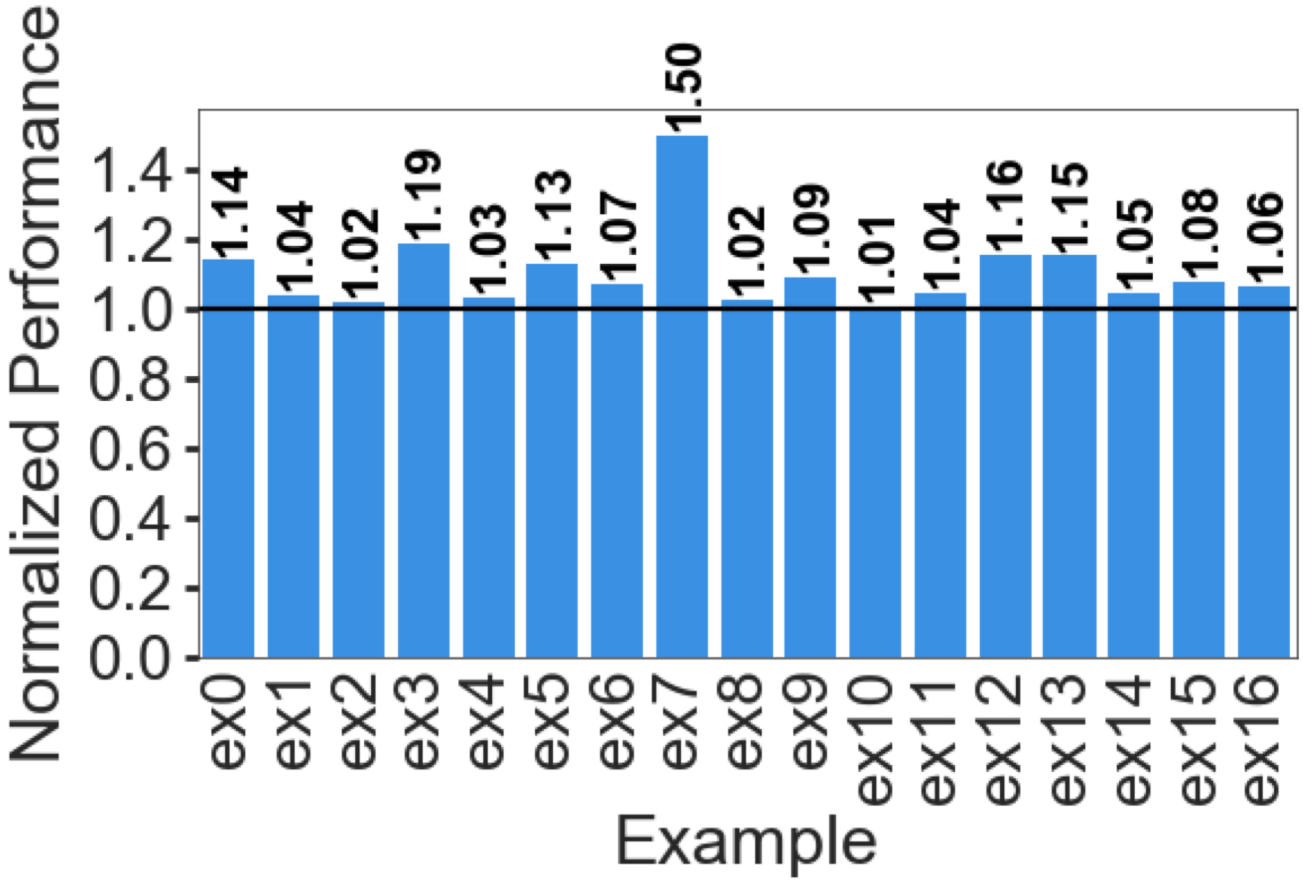
Motivation – Dot Product

27/35 of the configurations are better than that predicted by the current LLVM cost model!



Motivation – LLVM benchmarks

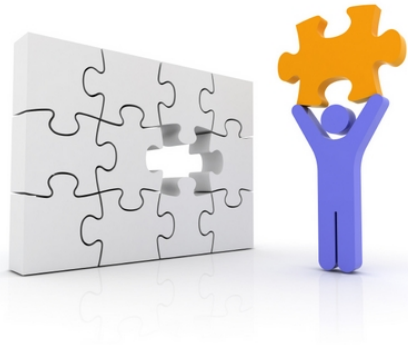
Brute-force search always outperforms the baseline cost model!



https://github.com/llvm/llvm-test-suite/tree/master/SingleSource/UnitTests/Vectorizer

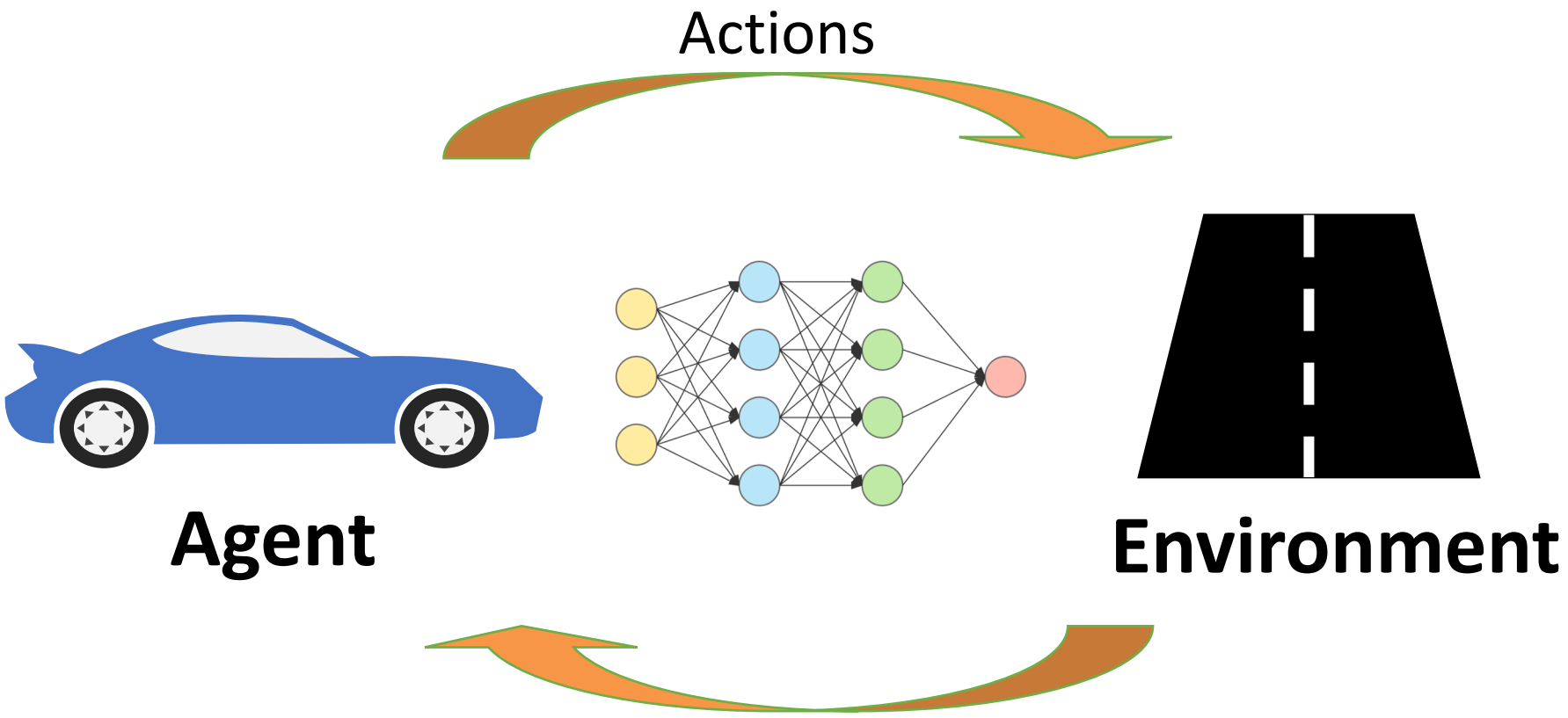
Our Contributions

- A framework that uses Deep RL for end to end vectorization
- Achieves 97% of the optimal performance
- Can generalize to different workloads
- A dataset of synthetic “for-loops”

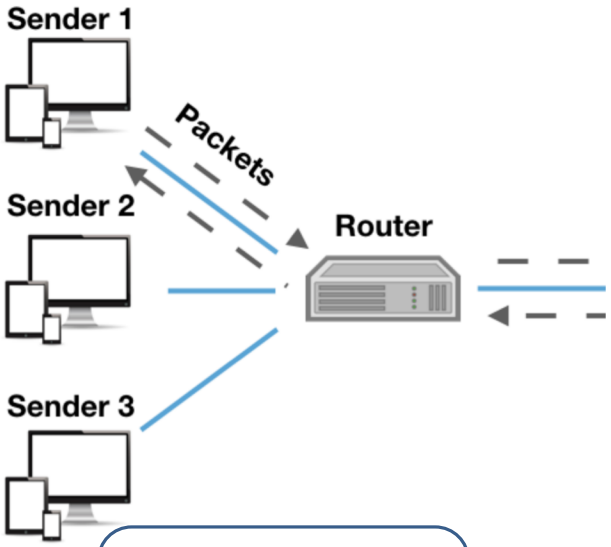


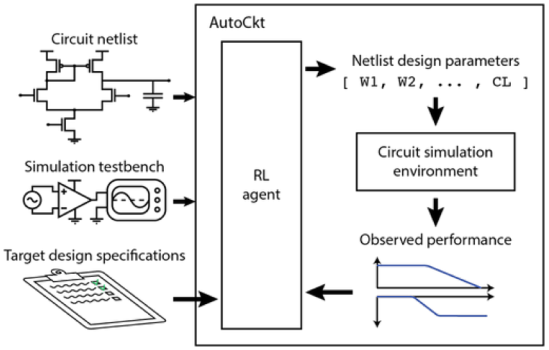
Ameer Haj-Ali, Nesreen K. Ahmed, Ted Willke, Yakun Sophia Shao, Krste Asanovic, and Ion Stoica. "NeuroVectorizer: End-to-End Vectorization with Deep Reinforcement Learning," In Proceedings of International Symposium on Code Generation and Optimization (CGO 2020, to appear), February 22–26, 2020, San Diego, CA, USA

Deep Reinforcement Learning



Deep RL is Used Everywhere in Systems





Which passes?
In which order?

gcc program.c -flag1 -flag2 ..

- **Ameer Haj-Ali**, Nesreen K. Ahmed, Ted Willke, Joseph Gonzalez, Krste Asanovic and Ion Stoica, "A View on Deep Reinforcement Learning in System Optimization", (submitted). Available on arXiv: https://arxiv.org/abs/1908.01275.
- **Ameer Haj-Ali**, Qijing Huang, William Moses, John Xiang, Ion Stoica, Krste Asanovic, John Wawrzynek, "Juggling HLS Phase Orderings in Random Forests with Deep Reinforcement Learning," Systems and Machine Learning (SysML 2020, to appear), March 2020.
- **Ameer Haj-Ali**, Qijing Huang, William Moses, John Xiang, Ion Stoica, Krste Asanovic, John Wawrzynek, "AutoPhase: Compiler Phase-Ordering for High-Level Synthesis with Deep Reinforcement Learning," The 27th IEEE International Symposium On Field-Programmable Custom Computing Machines (FCCM 2019), April 2019.
- Keertana Settaluri, **Ameer Haj-Ali**, Qijing Huang, Suhong Moon, Kourosh Hakhamaneshi, Ion Stoica, Krste Asanovic, Borivoje Nikolic, "AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs," Design, Automation & Test in Europe Conference & Exhibition (DATE 2020, to appear), March 2020.

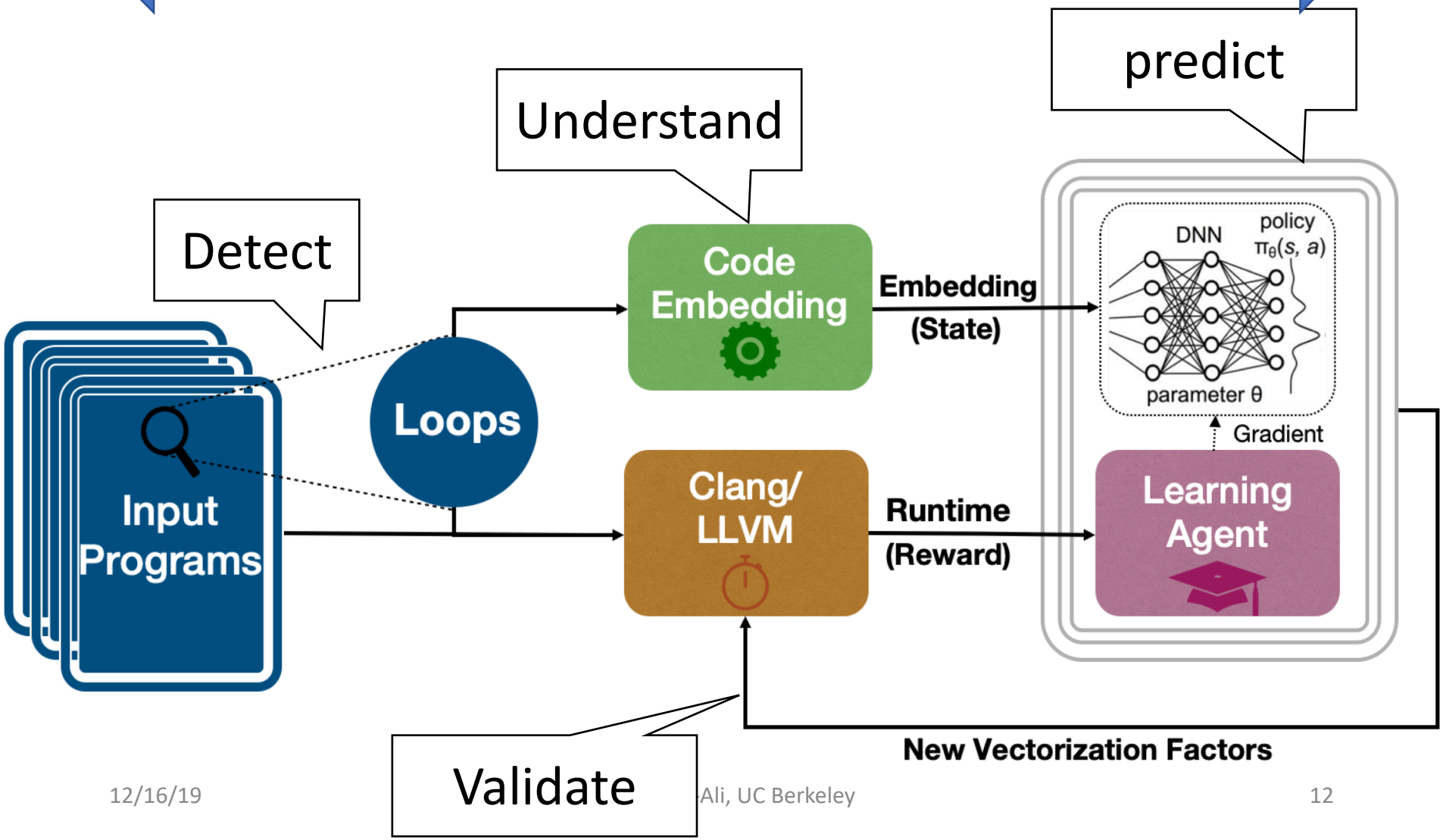
Why Vectorize with Deep RL?

- Unlike supervised learning → does not require labels
- Can learn complex structures and observation spaces
- Achieves 97% of the oracle performance
- Potential for more simultaneous optimizations
- Can optimize end-to-end

Solution Based on a Human Expert

- Detect the loops
- Understand the behavior (e.g., access patterns)
- Make a decision/prediction
- Validate

End-to-End Vectorization with Deep RL



Other possible embeddings

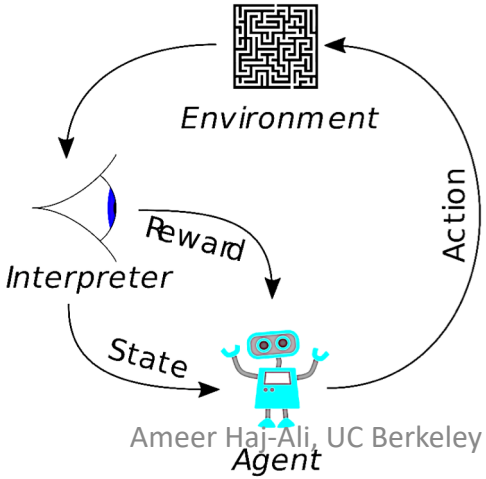
- Hand engineered features such as loop bounds
  - ✓Simple
  - ✓Understandable
  - —Error prone
  - —Requires experience
  - —Not comprehensive
  - —Does not capture access patterns
- Loop polyhedral representation
  - ✓Captures access patterns
  - ✓Works well in many cases
  - —Does not work well with predication
  - —Does not capture instruction distribution

Formulating the RL Environment

- State: code embedding of loops/code text
- Action: $VF \in [2^0, 2^1, 2^2, \ldots, \text{MAX\_VF}]$,

$$IF \in [2^0, 2^1, 2^2, \ldots, \text{MAX\_IF}],$$

- Reward: performance improvement normalized to the baseline; $reward = (t_{baseline} - t_{RL})/t_{baseline}$



What RL Does Exactly?

```
int vec[512] __attribute__((aligned(16)));
__attribute__((noinline))
int example1 () {
    int sum = 0;
    for(int i = 0; i<512; i++){
        sum += vec[i]*vec[i];
    }
    return sum;
}
```

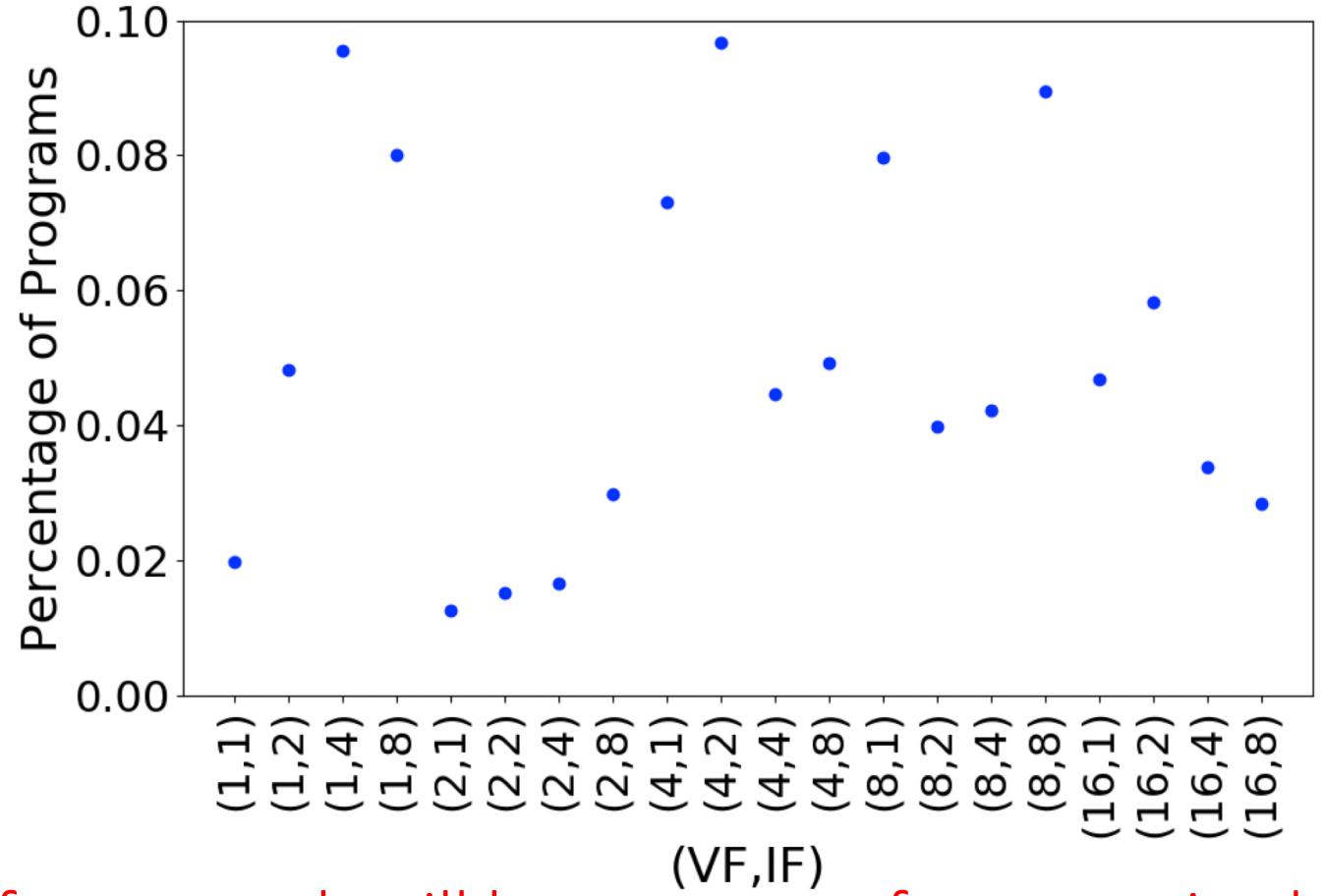
RL Agent's Action

```
int vec[512] __attribute__((aligned(16)));
__attribute__((noinline))
int example1 () {
    int sum = 0;
    #pragma clang loop vectorize_width(64)\\
    interleave_count(8)
    for(int i = 0; i<512; i++){
        sum += vec[i]*vec[i];
    }
    return sum;
}
```

Why Build a Synthetic Loops Dataset

- Lack of available benchmarks
- The alternative is to vectorize a single loop at a time and recompile the entire benchmark
  - Took days of training on the SPEC CPU 2017 benchmarks
    - Long compilation time
    - Long execution times
  - Hard to isolate speedups of loops
  - Not enough loops for training

Distribution of the Optimal Factors in the Dataset



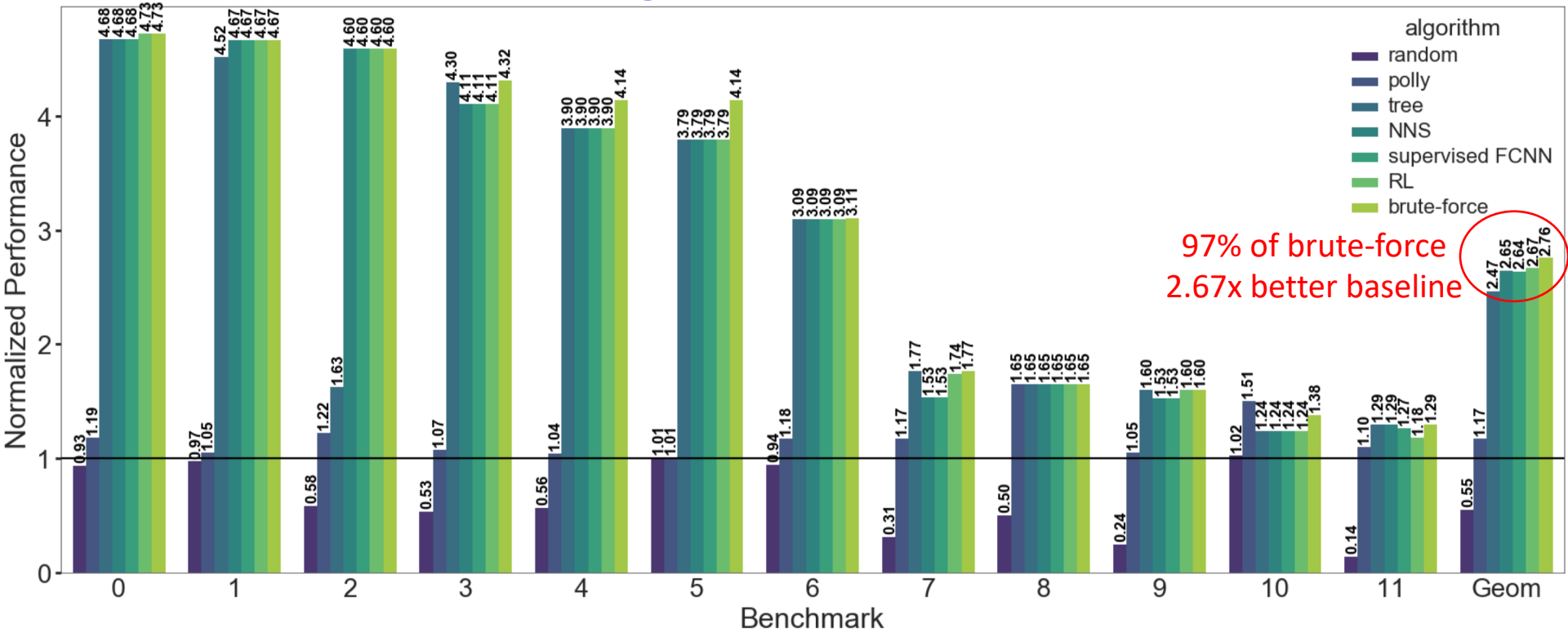
Brute-force search will be necessary for supervised learning!

Evaluation

- RL algorithm: proximal policy optimization
- RL versus (1) Polly, (2) decision trees, (3) nearest neighbor search, (4) supervised FCNN, (5) baseline cost model in LLVM, and (6) brute-force search
- Trained only on the synthetic dataset (~5k samples)
- Inference on PolyBench, MiBench, LLVM Benchmarks

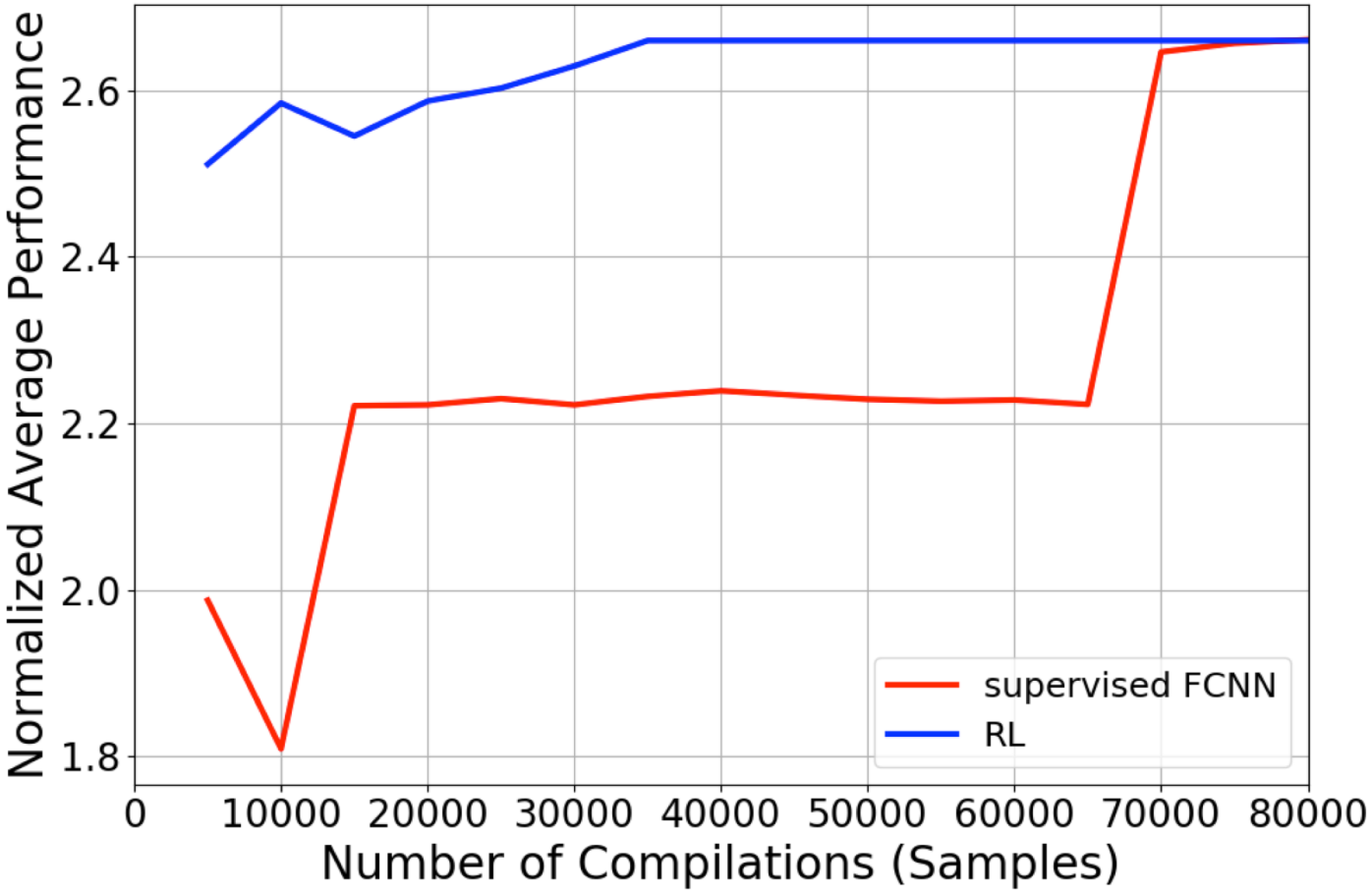
Results – LLVM Benchmarks

(Higher is better)

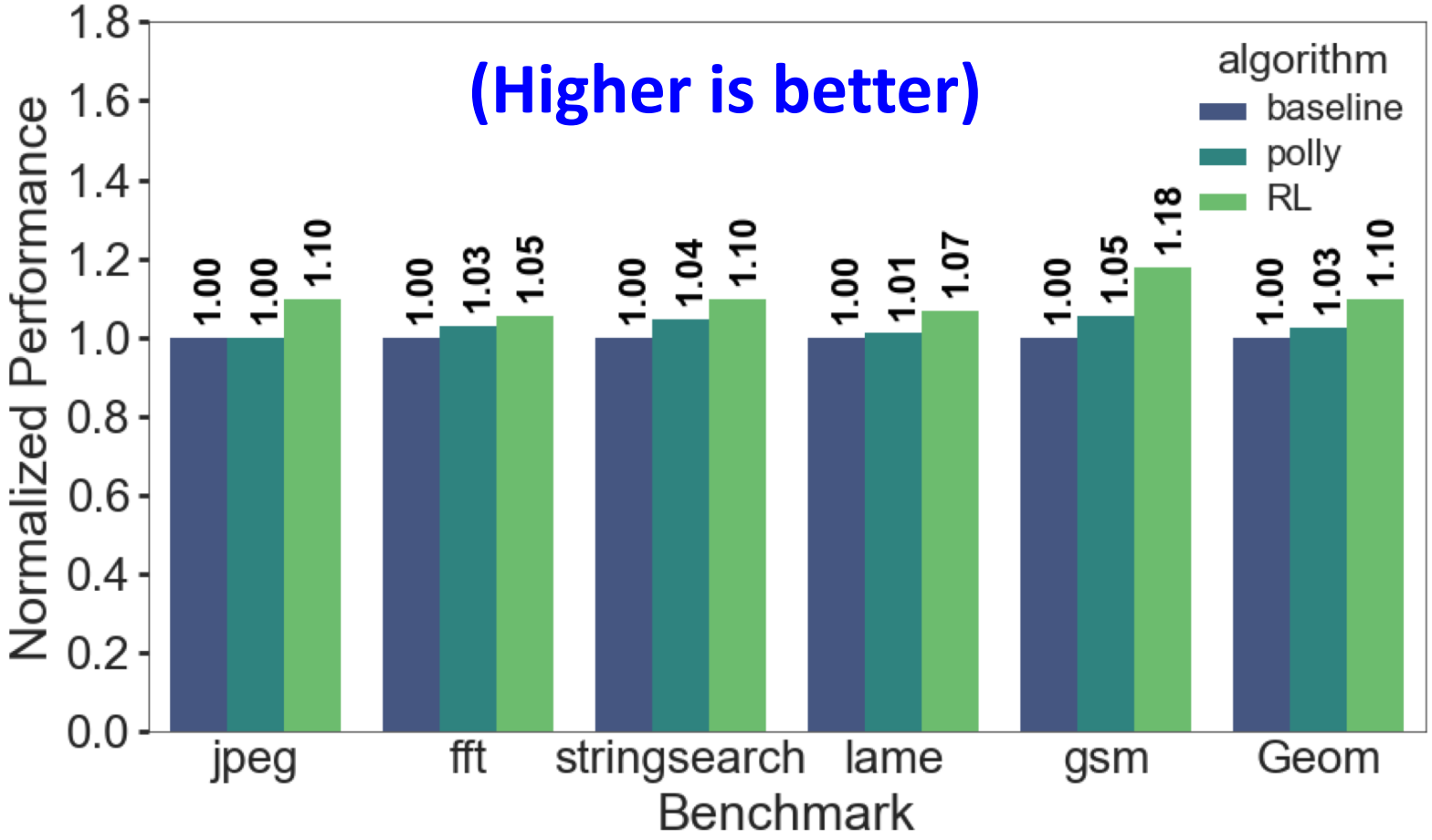


Results – Performance/Number of Compilations

RL requires 14X less compilations.



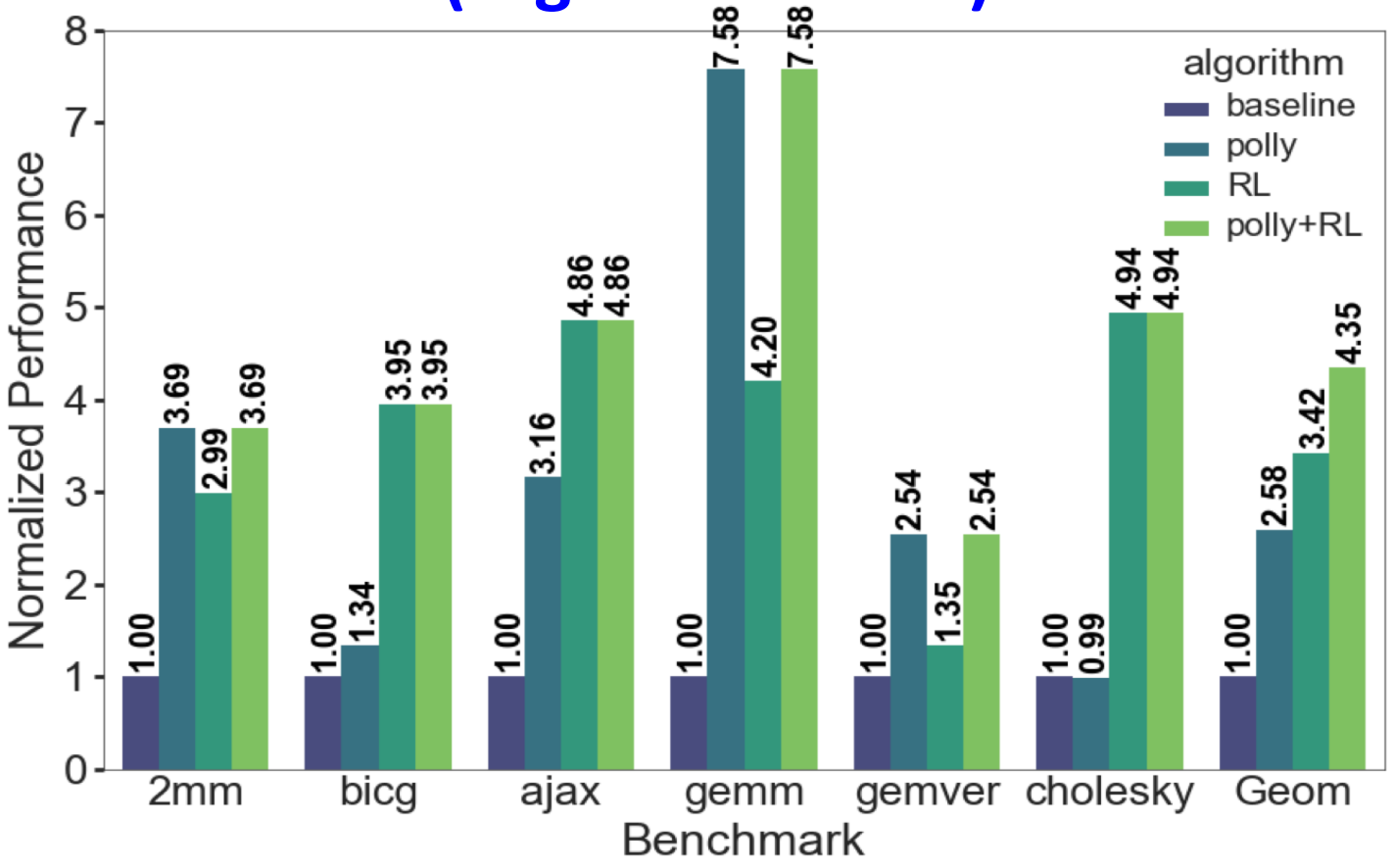
Transfer Learning - MiBench



Transfer Learning – PolyBench (Average)

1.33× better than polly and 3.42× better than –O3

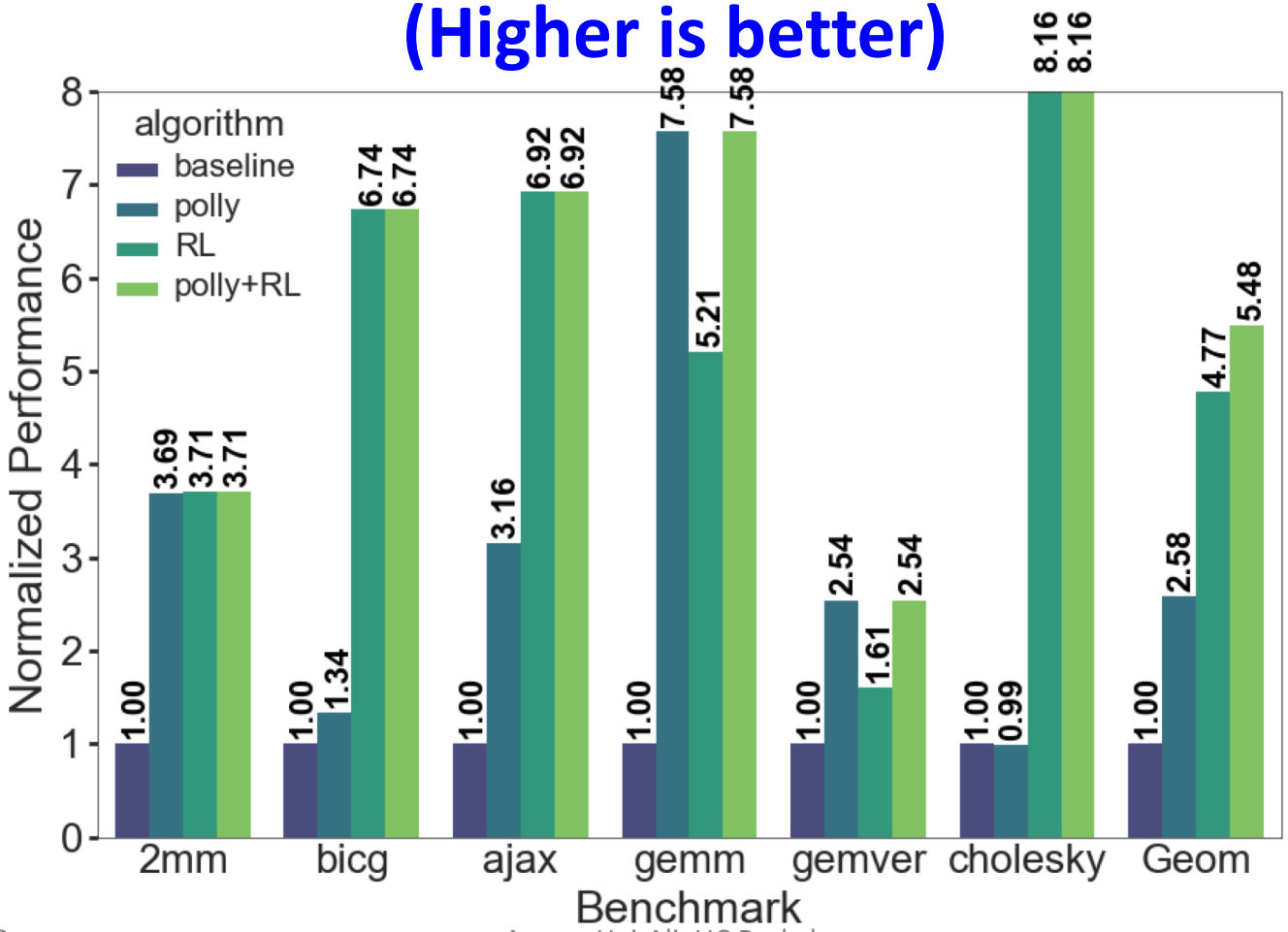
(Higher is better)



Transfer Learning – PolyBench (Best)

1.85× better than polly and 4.77× better than –O3

(Higher is better)



More in the Paper …

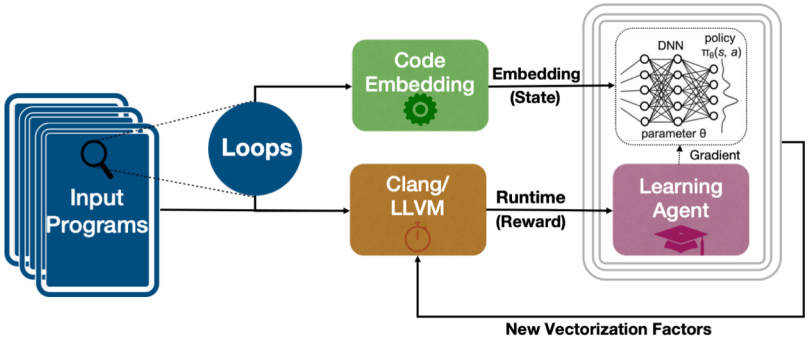
Ameer Haj-Ali, Nesreen K. Ahmed, Ted Willke, Yakun Sophia Shao, Krste Asanovic, and Ion Stoica. "NeuroVectorizer: End-to-End Vectorization with Deep Reinforcement Learning," In Proceedings of International Symposium on Code Generation and Optimization (CGO 2020, to appear), February 22–26, 2020, San Diego, CA, USA

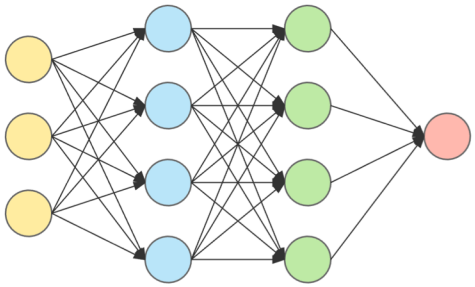
Available on arXiv: https://arxiv.org/abs/1909.13639

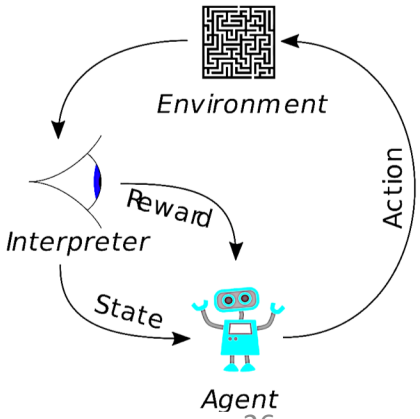
Conclusion

Deep RL Can …

- Automatically vectorize loops and tune compiler pragmas
- Achieve 97% the optimal performance in vectorization
- Generalize to different workloads
- Co-optimize multiple pragmas/objectives







Acknowledgement

- Nesreen K. Ahmed, Ted Willke, Yakun Sophia Shao, Krste Asanovic, and Ion Stoica

- Ronny Ronen, Ayal Zaks, Gadi Haber, Hideki Saito, Pankaj Chawla, Andrew Kaylor and anonymous reviewers for their insightful feedback and suggestions

Thanks

https://github.com/Intel-Academic/NeuroVectorizer

Ameer Haj-Ali
ameerh@berkeley.edu