**Background**

“Program testing can be used to show the presence of bugs, but never to show their absence!”
— Edsger W. Dijkstra

**Program Verification**

“If $P$ holds before executing $S$, then $Q$ holds afterwards.”

How to prove $\{P\} S \{Q\}$?

- Proving $\{P\} S \{Q\}$ requires deep logical reasoning
- Finding $I$ is a fundamental problem in program verification
- Traditionally, ad hoc features are used to find $I$
- Challenge problem in Artificial Intelligence

**Challenges**

How to represent programs so that a neural policy can be learned?

How to predict loop invariant based on the neural representation?

How to learn a policy from an automated theorem prover?

**Representing programs using ASTs and Dataflow**

**Experiments**

Collected 133 benchmark programs

Compared our framework Code2Inv with 3 SOTA approaches

Code and data: [https://github.com/PL-ML/code2inv](https://github.com/PL-ML/code2inv)

**Higher performance even without training**

**Attention and CE improve performance**

**Successfully generalizes to new programs**

**Visualizing attention over code**