# EditLord: Learning Code Transformation Rules for Code Editing

Weichen Li, Albert Jan, Baishakhi Ray, Junfeng Yang, Chengzhi Mao, Kexin Pei

THE UNIVERSITY OF CHICAGO · COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK · RUTGERS

ICML International Conference On Machine Learning

## Introduction

### Goal of Code Editing

Introduce desired code property changes without changing the original code's intended functionality.

### Limitation of Existing Approaches

Formulate code editing as an implicit end-to-end task

- internalize code editing rules as model weights
- ignoring code transformations

### Direct LLM editing does not work well

Count and print how many of the given heights are greater than or equal to a specified threshold.

```
Sample Input:   Sample Output:
5 10            2
2 4 15 9 23
```

**Pre-Edit Code**

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int n, k;
    cin >> n >> k;
    vector<int> h(n);
    for (int i = 0; i < n; i++) cin >> h[i];
    sort(h.begin(), h.end(), greater<int>());
    int cnt = 0;
    for (int i = 0; i < n; i++)
        if (h[i] >= k) cnt++;
    cout << cnt << endl;
    return 0;
}
```

**GPT-4 End-to-End Edited Code**

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int n, k;
    cin >> n >> k;          // keeps using cin/cout
    vector<int> h(n);
    for (int i = 0; i < n; i++) cin >> h[i];   // keeps unnecessary code
    sort(h.begin(), h.end(), greater<int>());
    int cnt = 0;
    for (int i = 0; i < n && h[i] < k; i++)
        cnt++;
    cout << cnt << endl;    // Attempted to optimize loop condition, but breaks the functionality
    return 0;
}
```

## Key Idea: LLM as Inductive Rule Learner for Code Editing Rules

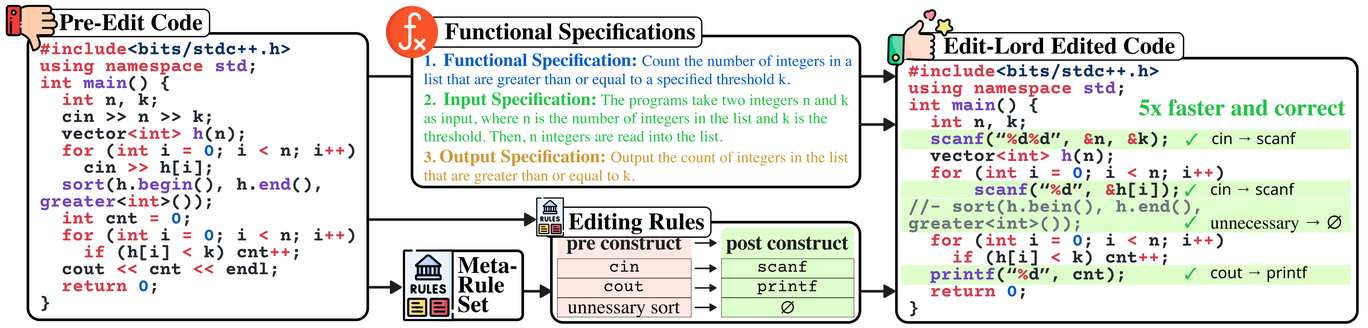
### Operate in a structured, discrete transformation space

Challenge 1: model doesn't know intended functionality

Solution 1: make model explicitly learn functional specifications

Challenge 2: model doesn't know transformation rules

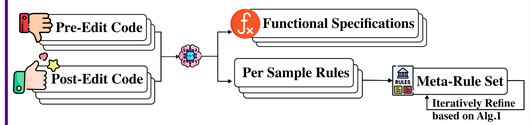
Solution 2: make model explicitly learn 1) set of transformation rules and 2) applicable rules for each code sample




P(Post | Pre)
= P(Func Spec | Pre)
* P(Rule | Pre, Func Spec)
* P(Post | Pre, Func Spec, Rule)

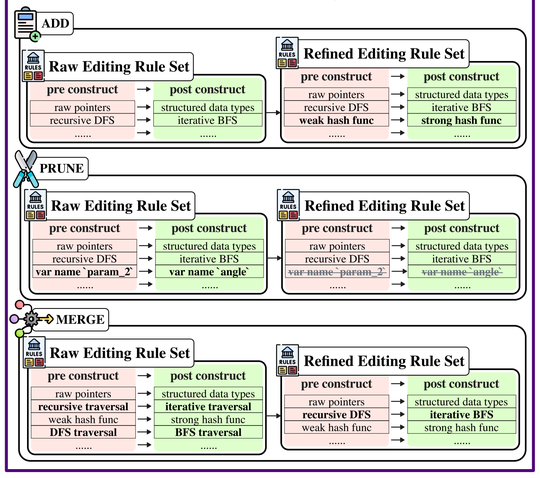
## Rule Learning

### LLM as Inductive Rule Learner

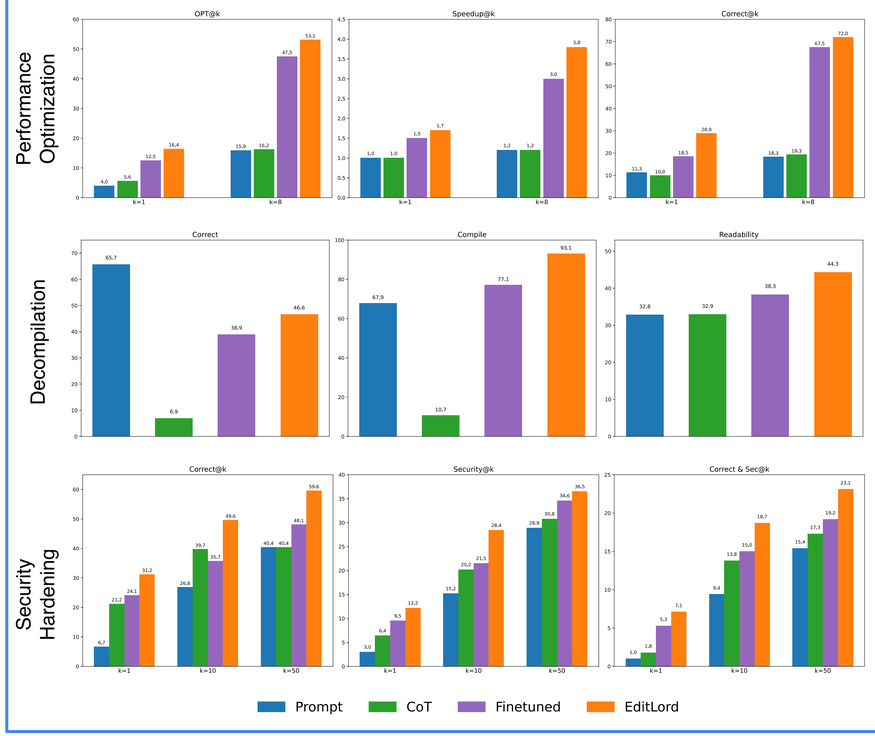



### Refinement Actions

3 simple actions to identify inductive rules




## Main Results




### Performance Opt.

- Input: slow code
- Output: fast code
- Metric: 1. #programs that becomes faster, 2. speedup, 3. correctness
- 23% faster

### Decompilation

- Input: unreadable Ghidra decompiled code
- Output: readable code
- Metric: 1. correctness, 2. compilability, 3. readability
- 12% more readable

### Secure Hardening

- Input: vulnerable code
- Output: secure code
- Metric: 1. security, 2. correctness, 3. both security and correctness
- 27% safer