

SmartOpt — Simple AI-Powered Code Optimizer

Overview

SmartOpt is a **prototype compiler optimization advisor** for **C, C++, and Rust** programs. It analyzes user-submitted code, compiles it with different optimization flags (like -O0, -O1, -O2, -O3, and -Os), and identifies which flag performs best based on **runtime, compile time, and binary size**.

This is a **simple, working prototype** — not a research-level mathematical optimizer. It focuses on **small code samples** and **basic loops, operations, or functions**, not on advanced workloads (like matrix multiplication, sin/cos, or heavy math libraries).

Current Capabilities

- Supports **C, C++, and Rust**
- Benchmarks compile time, runtime, and binary size
- Provides **best optimization flag**
- Uses a simple **ML model** (RandomForest/XGBoost) for prediction
- Automated deployment via **Jenkins + Cloud Run**

Limitations (for prototype phase)

- Designed for **simple code snippets** only
- No mathematical or high-complexity workloads (e.g., sin, cos, matrix math)
- Binary sizes may appear similar for small programs
- Focused on functionality, not precision modeling
- Optional **LLM explanation** (Gemma-2B) for plain-language insight(Working on this)

Future Enhancements

- Add support for **mathematical and scientific workloads**
- Improve ML model using **code-level feature extraction**
- Introduce **GNN or RL-based learning** for better predictions
- Add advanced benchmarking and metrics visualization
- Integrate more languages and improve frontend UX

Purpose

The goal of this version is to **demonstrate concept and workflow** — how DevOps automation (Jenkins + GCP) and Machine Learning can work together in compiler optimization.

It's not about mathematical depth — it's about building a **working, cloud-based AI compiler advisor**.

Sample Output :

Rust Outputs

Using Docker :

```
{"filename":"tmpqpvui5_va.rs","language":".rs","best_flag": "-O1","flags":[{"flag": "-O0","compile_time":0.26019,"runtime":0.00061,"binary_size":3916608,"status":"ok"}, {"flag": "-O1","compile_time":0.16952,"runtime":0.00058,"binary_size":3915952,"status":"ok"}, {"flag": "-O2","compile_time":0.13158,"runtime":0.00059,"binary_size":3915952,"status":"ok"}, {"flag": "-O3","compile_time":0.121,"runtime":0.00054,"binary_size":3915952,"status":"ok"}, {"flag": "-Os","compile_time":0.12937,"runtime":0.00052,"binary_size":3915952,"status":"ok"}],"explanation": "⚠ Could not extract explanation from model."}%
```

Using UI :

SmartOpt – AI-Powered Compiler Optimization

Paste your C / C++ / Rust code below:

```
<> Paste your code here
1 fn main() {
2     let mut sum: u64 = 0;
3     for i in 1..=1_000_000 {
4         sum += i;
5     }
6     println!("Sum of first 1 million numbers: {}", sum);
7 }
```

Analyze Optimization

✓ Best Optimization Flag: -O3

Optimization Metrics

Flag	Compile Time	Runtime	Binary Size	Status
-O0	1.98873	0.01359	3900192	ok
-O1	0.48423	0.00525	3896968	ok
-O2	0.37965	0.00457	3896936	ok
-O3	0.3872	0.00472	3896936	ok
-Os	0.38816	0.00524	3896904	ok

⚠ Could not extract explanation from model.

C++ Outputs

Using Docker :

```
"filename":"tmpk5guz6bg.cpp","language": ".cpp", "best_flag": "-O1", "flags": [{"flag": "-O0", "compile_time": 0.22146, "runtime": 0.00075, "binary_size": 70600, "status": "ok"}, {"flag": "-O1", "compile_time": 0.22051, "runtime": 0.00073, "binary_size": 70600, "status": "ok"}, {"flag": "-O2", "compile_time": 0.20621, "runtime": 0.0007, "binary_size": 70600, "status": "ok"}, {"flag": "-O3", "compile_time": 0.20739, "runtime": 0.00069, "binary_size": 70600, "status": "ok"}, {"flag": "-Os", "compile_time": 0.20756, "runtime": 0.00067, "binary_size": 70600, "status": "ok"}], "explanation": "⚠ Could not extract explanation from model."}%
```

Using UI:

SmartOpt – AI-Powered Compiler Optimization

Paste your C / C++ / Rust code below:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     long long total = 0;
6     for (int i = 1; i <= 1000000; i++) {
7         total += i;
8     }
9     cout << "Sum of first 1 million numbers: " << total << endl;
10    return 0;
11}
12
```

Analyze Optimization

✓ Best Optimization Flag: -O1

Optimization Metrics

Flag	Compile Time	Runtime	Binary Size	Status
-O0	0.88936	0.00709	16376	ok
-O1	0.85161	0.01343	16520	ok
-O2	0.81783	0.00408	16520	ok
-O3	0.80227	0.00509	16520	ok
-Os	0.82725	0.00455	16464	ok

C Outputs:

Using Docker:

```
{"filename":"tmpjb4xqts2.c","language":"c","best_flag": "-O1","flags":[{"flag": "-O0","compile_time":0.05148,"runtime":0.00041,"binary_size":70432,"status":"ok"}, {"flag": "-O1","compile_time":0.04576,"runtime":0.00036,"binary_size":70432,"status":"ok"}, {"flag": "-O2","compile_time":0.02968,"runtime":0.00036,"binary_size":70432,"status":"ok"}, {"flag": "-O3","compile_time":0.03317,"runtime":0.00036,"binary_size":70432,"status":"ok"}, {"flag": "-Os","compile_time":0.03471,"runtime":0.00045,"binary_size":70432,"status":"ok"}],"explanation":"⚠ Could not extract explanation from model."}%
```

Using UI :

SmartOpt — AI-Powered Compiler Optimization

Paste your C / C++ / Rust code below:

```
1 #include <stdio.h>
2
3 int main() {
4     long long total = 0;
5     for (int i = 1; i <= 1000000; i++) {
6         total += i;
7     }
8     printf("Sum of first 1 million numbers: %lld\n", total);
9     return 0;
10}
11
```

Analyze Optimization

✓ Best Optimization Flag: -O1

Optimization Metrics

Flag	Compile Time	Runtime	Binary Size	Status
-O0	0.15036	0.00504	16000	ok
-O1	0.15479	0.00308	16000	ok
-O2	0.17227	0.00353	16000	ok
-O3	0.18455	0.00337	16000	ok
-Os	0.16634	0.00332	16000	ok

⚠ Could not extract explanation from model.