香港城市大學(東莞)
City University of Hong Kong
(Dongguan)

Development of a VS Code Plugin for Anomaly Log Parsing Based on LogLLM

Wenhao Li, Simin Xian, Yu Fu, Shuang Leng, Zhihong Zhang, Jiajun Zhang

GitCommit

Motivation

➢ **Problem Background**
System logs are essential for debugging, but their increasing volume and complexity make manual analysis inside the IDE inefficient.

➢ **Pain Points**
- **Weak Parsing**: Poor handling of complex log structures.
- **Privacy Exposure**: Cloud tools send sensitive data externally.
- **Workflow Break**: Constant switching to external platforms disrupts focus.

➢ **Our Goal**
To build a **local, smart VS Code plugin** that:
- Accurately parses logs using LogLLM.
- Masks private data with regex.
- Recommends solutions and enables in-IDE Q&A for seamless debugging.

Innovation

➢ **Novelty**
- **Hybrid BERT-Llama Model**: Uses a projector layer to align semantic encodings, enabling deep log understanding within memory limits.
- **IDE-Native & Local**: Runs entirely inside VS Code; no sensitive data leaves the local machine.
- **Regex-Powered Preprocessing**: Replaces variables with placeholders, preserving semantics without reliance on fragile log parsers.

➢ **Differentiation**
- **vs. Traditional Tools**: Ours uses full semantic LLM analysis instead of template-based parsing.
- **vs. Cloud LLM Services**: Ours is fully local and private; theirs send logs externally.
- **vs. General LLM Log Tools**: Ours is fine-tuned for logs and integrated directly into the developer's IDE workflow.

System Architecture & Technical Design

The plugin enables a one-stop, local log analysis workflow inside VS Code: upload logs → automatic privacy masking → anomaly detection by the LogLLM model **(BERT+Llama+Projector)** → results and solutions displayed in an interactive sidebar.

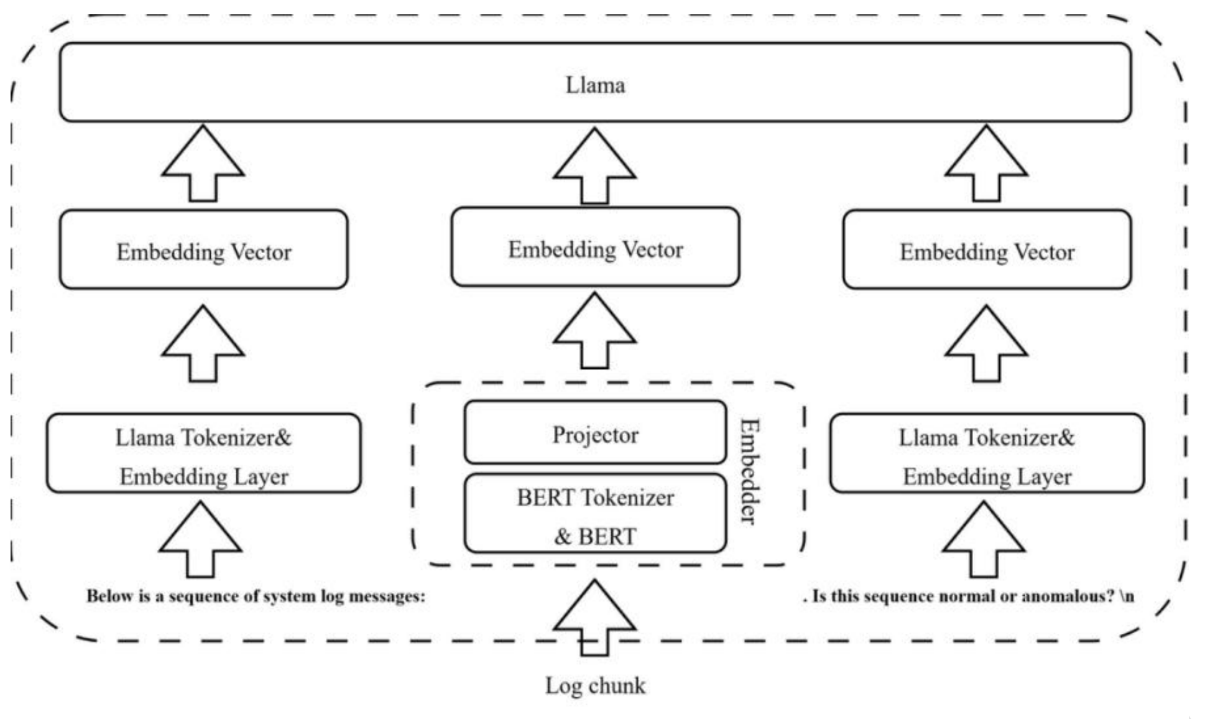
Llama
Embedding Vector
Embedding Vector
Embedding Vector
Llama Tokenizer& Embedding Layer
Projector
BERT Tokenizer & BERT
Embedder
Llama Tokenizer& Embedding Layer
Below is a sequence of system log messages:
. Is this sequence normal or anomalous? \n
Log chunk

Fig1: Framework of LogLLM

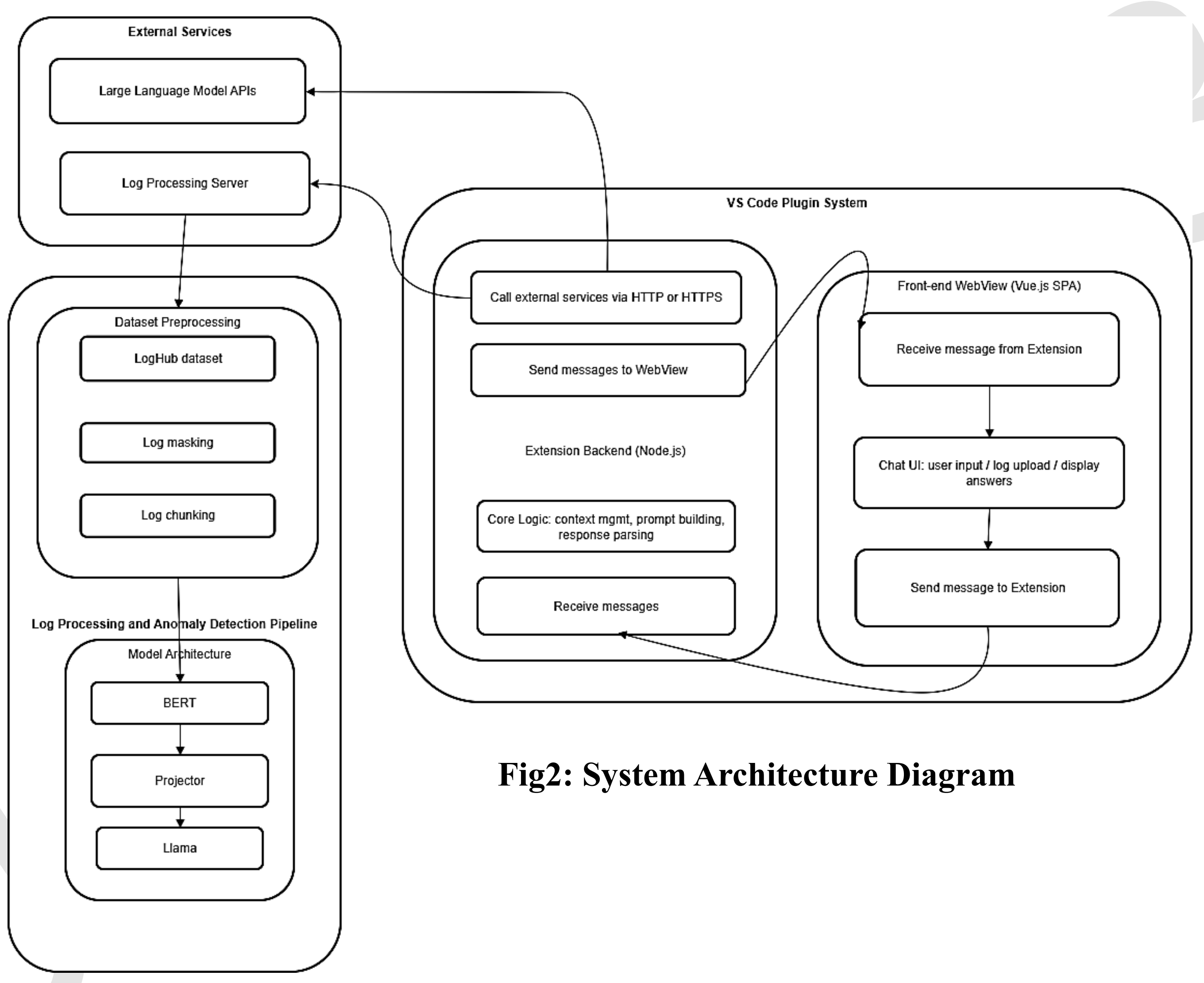
External Services
Large Language Model APIs
Log Processing Server
Dataset Preprocessing
LogHub dataset
Log masking
Log chunking
Log Processing and Anomaly Detection Pipeline
Model Architecture
BERT
Projector
Llama
VS Code Plugin System
Call external services via HTTP or HTTPS
Send messages to WebView
Extension Backend (Node.js)
Core Logic: context mgmt, prompt building, response parsing
Receive messages
Front-end WebView (Vue.js SPA)
Receive message from Extension
Chat UI: user input / log upload / display answers
Send message to Extension

Fig2: System Architecture Diagram

➢ **Technical Details:**
- Employs regular expressions for privacy filtering.
- Uses a hybrid LLM architecture for semantic understanding.
- Features IDE-native integration via a WebView + Node.js extension.

➢ **Non-Functional Attributes:**
The design ensures: security & privacy through local execution; maintainability via an MVC frontend/backend separation; and high performance via window-based chunking for large-file processing.

Software Process & Management

➢ **Sprint:**
- **Sprint 1:** Requirements Specification Document, System Architecture Diagram, User Stories & Sprint 2 Backlog, Configured development environment.
- **Sprint 2:** Runnable VS Code plugin prototype, Backend service with integrated LogLLM, Frontend-backend communication APIs.
- **Sprint 3:** Production-ready plugin package, Automated test suite & final test report, Complete project documentation, Performance optimization report.

➢ **Collaboration:**
We used GitHub for version control, task management, story point estimation, sprint tracking and collaboration

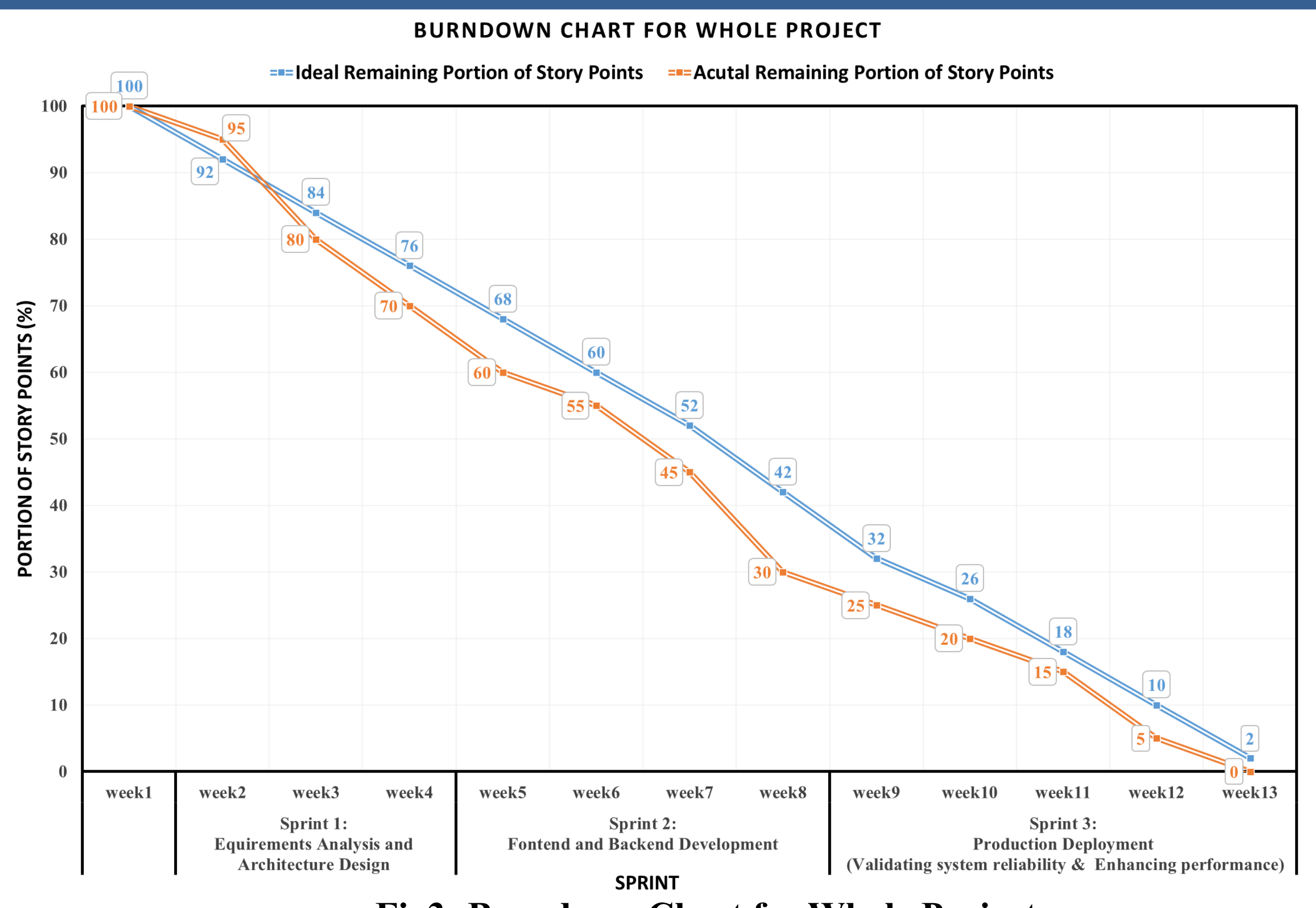

Fig3: Burndown Chart for Whole Project

Conclusion & Demo Presentation

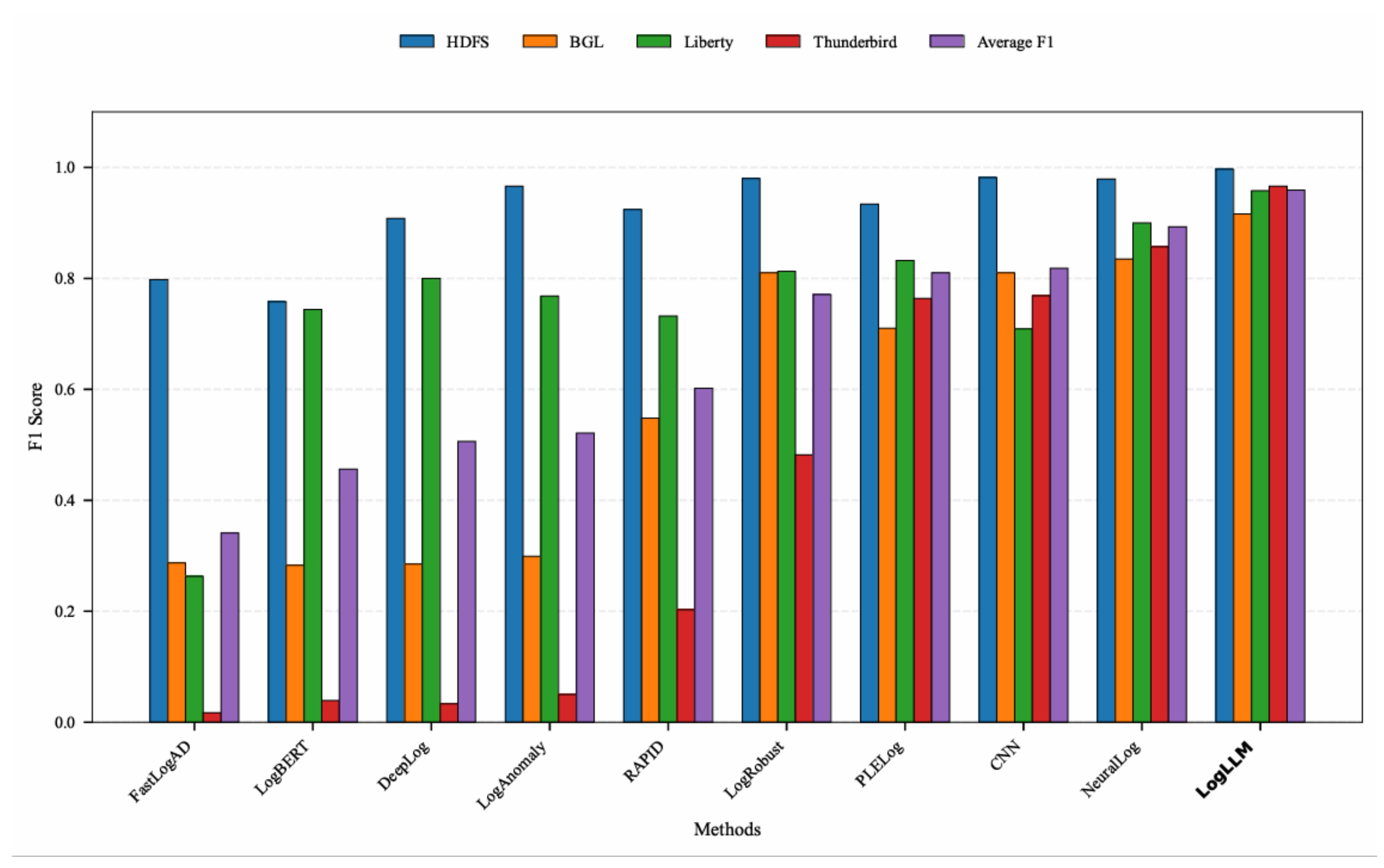

Fig4: Bar chart comparing average F1-scores of anomaly detection methods

➢ **Results:**
- The plugin effectively solves the core problems, outperforming existing tools with:
- **Highest accuracy: 0.959 average F1-score**, outperforming peers by 6.6%.
- **Complete functionality:** Full anomaly detection, solution ranking, and 100% privacy masking.
- **Proven robustness:** Stable under **200+ concurrent users** (>97% success rate).

➢ **Conclusion:**
- We delivered a production-ready, local VS Code plugin that makes log analysis seamless and secure within the IDE.
- Future work will enable custom model fine-tuning and cross-file anomaly tracing.

➢ **Demo:** [Scan QR Code for Video/GitHub].