Inferno Tactics

1st Shaurya Mathur
Dept. of Computer Science & Engineering
University at Buffalo
Buffalo, USA
smathur4@buffalo.edu

2nd Shreyas Bellary Manjunath
Dept. of Computer Science & Engineering
University at Buffalo
Buffalo, USA
sbellary@buffalo.edu

Abstract—In this project, Inferno Tactics, we explore the integration of Reinforcement Learning (RL) with Deep Learning (DL) to develop a proactive system for wildfire prediction and verification using satellite imagery and spatio-temporal data. The system is structured into two interconnected modules. The first module leverages a deep learning model trained on historical wildfire patterns and Earth Engine satellite data to forecast potential wildfire events — predicting both the likely date and geographic location of ignition. Building on this, the second module introduces a reinforcement learning agent that interacts with a dynamic environment of evolving satellite data. The agent's objective is to validate the predicted wildfire events by actively selecting and analyzing recent image sequences, maximizing verification accuracy through feedback-driven exploration. This multi-stage architecture harnesses the predictive power of DL and the decision-making strengths of RL to create a more robust and intelligent wildfire monitoring pipeline. While still under active development, preliminary results suggest that the RL-enhanced validation significantly improves the system's ability to filter out false positives and deliver timely alerts for disaster management.

Index Terms—wildfire, gym, deep learning, reinforcement learning

I. Introduction

Wildfires are increasingly becoming a global threat, with rising temperatures and shifting climate patterns contributing to their frequency and intensity. These fires not only devastate ecosystems but also endanger human lives and infrastructure. Traditional methods of wildfire detection and response often rely on environmental monitoring and sensor-based data; however, these approaches are largely reactive and struggle to provide timely and strategic interventions over vast, unmonitored regions.

Inferno Tactics explores a novel approach that integrates Deep Learning (DL) with Reinforcement Learning (RL) to both predict and plan for wildfire events. The DL component analyzes satellite imagery and historical wildfire data to forecast the time and location of potential wildfire outbreaks. In parallel, the RL component simulates a dynamic environment where wildfires may occur, allowing an intelligent agent to learn optimal mitigation strategies through trial-and-error interactions. These strategies can involve decisions such as resource deployment, early evacuation planning, or the prioritization of firebreak construction.

By modeling wildfire scenarios as a Markov Decision Process (MDP), the RL agent learns to make sequential decisions that aim to minimize long-term damage. This dual-system framework—combining predictive modeling with adaptive planning—seeks to enable a more proactive and automated wildfire management system.

This report presents the implementation and results of the project, including detailed descriptions of the geospatial data acquisition pipeline, simulation environment, and reinforcement learning training framework. We demonstrate the feasibility of combining DL and RL for improving wildfire prediction, verification, and strategic response planning.

II. Background and Motivation

A. Wildfires and the Need for Intelligent Response

Wildfires are becoming increasingly destructive due to climate change, land-use patterns, and prolonged dry seasons. Traditional wildfire response mechanisms, while valuable, often rely on reactive strategies that lack foresight and scalability. In the face of rapidly spreading fires, there's a growing need for proactive and intelligent systems that not only predict the likelihood of wildfires but also simulate and learn effective mitigation responses in real-time.

B. Simulating Wildfire Behavior with Real-World Data

To study wildfire dynamics in a realistic setting, we have developed an interactive simulation environment using React.js and three.js. This simulation serves as a visual and behavioral model of wildfire propagation, incorporating real-world geographic data to enhance its accuracy.

Specifically, we use Google Earth Engine APIs to fetch terrain elevation and land cover data, which are then integrated into the simulation. These attributes—along with wind speed and direction—play a crucial role in shaping how fire spreads. By grounding the simulation in actual Earth data, we aim to mimic plausible wildfire scenarios that are both visually intuitive and spatially accurate.

C. Modeling Firefighting Agents via Reinforcement Learning

At the core of our project is a Reinforcement Learning (RL) agent designed to develop and optimize wildfire

mitigation strategies. We connect our React-based frontend to a custom Python-based Gym environment via WebSockets, enabling real-time interaction between the simulation and the RL backend.
The initial agent we have developed simulates a helitack unit—a helicopter-based firefighting strategy. Using the Stable Baselines3 library, we train the agent to explore and learn optimal policies to suppress fires efficiently across different terrain and wind conditions. The agent's goal is to minimize the spread and damage caused by the fire over time, adapting its actions based on changing environmental dynamics.

D. Motivation for Hierarchical Multi-Agent Planning

While the helitack strategy is effective in certain scenarios, real-world wildfire mitigation often involves a combination of approaches, including fire trucks, aerial water drops, and firebreak construction. To simulate such complexity, we plan to extend our system using Hierarchical Reinforcement Learning (HRL).
In this framework, high-level policies will determine which mitigation strategy (e.g., helitack, fire truck dispatch) to deploy, while lower-level agents execute specialized tasks based on the chosen strategy. This modular architecture enables coordinated planning, resource allocation, and scalable learning across different fire response techniques. Our motivation stems from building a generalized, adaptable simulation tool that can train agents capable of multi-modal disaster response. By modeling realistic wildfire environments and training agents through trial-and-error, we aim to develop an intelligent system that can assist human decision-makers in planning efficient, timely, and context-aware fire mitigation strategies.

III. Geodata Extraction and Processing

A critical component of our system is the ability to model realistic wildfire scenarios using actual geographical data. To achieve this, we developed a comprehensive geodata extraction pipeline that sources, processes, and transforms Earth observation data into simulation-ready assets.

A. Google Earth Engine Integration

We implemented a Python module that interfaces with the Google Earth Engine API to extract two primary types of data:

- Elevation Data: Digital Elevation Models (DEMs) from the Shuttle Radar Topography Mission (SRTM) dataset, providing terrain height information at 30-meter resolution globally.
- Land Cover Data: ESA WorldCover 2020 dataset, which classifies land into 11 distinct categories including forests, grasslands, urban areas, and water bodies.

The extraction process begins with a set of predicted wildfire ignition coordinates (latitude and longitude) obtained from our deep learning prediction module. These coordinates serve as the center point for a regional extraction, typically covering a 10km $\times$ 10km area.

B. Data Processing Pipeline

Our geodata processing workflow consists of several stages:

1) API Request and Authentication: Secure authentication with Earth Engine servers and formulation of data retrieval requests based on geographical bounds.
2) Resolution Harmonization: Resampling of disparate data sources to ensure consistent spatial resolution across datasets.
3) GeoTIFF Generation: Export of processed data as GeoTIFF files, preserving both the raw data values and geospatial metadata.
4) Heightmap Conversion: Transformation of elevation GeoTIFFs into grayscale heightmap PNG images, where pixel intensity corresponds to elevation.
5) Land Cover Reclassification: Mapping of ESA WorldCover classes to simulation-relevant fire behavior parameters (e.g., fuel load, ignition probability).

Fig. 1. Heightmap generated from SRTM elevation data, with pixel intensity representing terrain elevation.

C. Coordinate System Transformation

To bridge the gap between geographic coordinates (latitude/longitude) and the simulation's Cartesian space, we implemented coordinate transformation utilities that:

- Convert between WGS84 geographic coordinates and local UTM projections.
- Map real-world distances to simulation units while preserving spatial relationships.
- Align elevation data with the visualization engine's height field requirements.

This careful transformation ensures that fire spread dynamics in the simulation accurately reflect how wildfires would propagate across the actual terrain, accounting for slope effects, fuel distribution, and topographical barriers.

D. Integration with Simulation Environment

The processed geodata serves as the foundational layer for our simulation environment:

- Heightmaps are used by Three.js to generate 3D terrain meshes with accurate elevation profiles.
- Land cover rasters inform the simulation's fuel model parameters, affecting fire intensity and spread rates.
- Additional derived layers (e.g., slope, aspect) are calculated to enhance fire behavior modeling.

By automating this entire pipeline, our system can quickly generate simulation environments for any wildfire-prone region globally, enabling both targeted training scenarios and rapid response simulations for newly predicted wildfire events.

IV. Simulation Environment Development

A. Three-Dimensional Visualization Framework

To provide an intuitive and realistic representation of wildfire dynamics, we developed a browser-based simulation environment using React.js for the application framework and Three.js for 3D graphics rendering. This environment serves dual purposes: (1) providing human operators with a visual interface to understand wildfire behavior and intervention effects, and (2) generating visual and numerical data to train reinforcement learning agents.

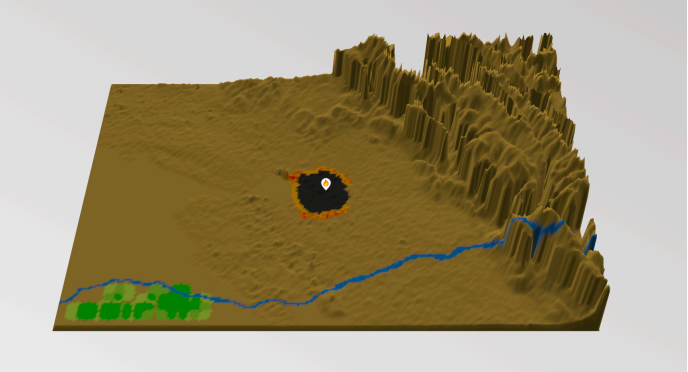


Fig. 2. Interactive 3D wildfire simulation environment developed in Three.js.

The visualization framework includes several key components:

- Terrain Renderer: Converts heightmap data into a detailed 3D mesh with accurate elevation profiles. The renderer applies texture mapping based on land cover classification to visually differentiate between forests, grasslands, urban areas, and other terrain types.
- Fire Propagation Visualizer: Implements a particle system and custom shaders to represent active fire fronts, burn intensity, and smoke dispersion. The visual intensity of fire correlates with the underlying fire behavior model's calculated heat release rate.
- Agent Visualization: Renders firefighting units (e.g., helicopters, ground crews) as interactive 3D models that respond to agent decisions in real-time. Movement paths and action effects (e.g., water drops) are visualized to provide interpretability of agent behavior.
- Environmental Controls: User interface elements allow manual adjustment of environmental parameters such as wind speed, wind direction, and fuel moisture to test different wildfire scenarios.

B. Fire Behavior Modeling

At the core of our simulation is a cellular automaton-based fire spread model that calculates fire propagation based on physical and environmental factors:

- Ignition Logic: Fire cells can transition from unburned to burning states based on proximity to active fire, fuel type, and stochastic factors representing ember transport.
- Spread Rate Calculation: The rate at which fire spreads from cell to cell is governed by the Rothermel surface fire spread model, adapted for our gridded environment. This model accounts for:
  - Fuel characteristics (type, load, density) derived from land cover data
  - Slope effects calculated from terrain elevation gradients
  - Wind vectors that influence directional spread
  - Atmospheric conditions including relative humidity and temperature
- Fuel Consumption: Each cell maintains a state representing available fuel, which depletes as the fire burns. This affects fire intensity and duration at specific locations.
- Fire Suppression Effects: The model incorporates the impact of firefighting actions, such as the cooling and fuel removal effects of water drops from helitack units.

C. Interactive Simulation Controls

To facilitate both human operation and agent training, we implemented a comprehensive control interface:

- Simulation Time Controls: Features for starting, pausing, accelerating, and resetting the simulation to support both real-time interaction and accelerated training.
- Camera Management: Multiple camera perspectives including top-down, first-person, and orbital views to observe fire behavior and agent actions from different angles.
- Scenario Editor: Tools for creating custom wildfire scenarios by placing ignition points, adjusting environmental conditions, and selecting pre-loaded terrain models.
- Data Visualization Overlays: Heat maps, vector fields, and other data visualization techniques to display

fire intensity, spread direction, and agent effectiveness metrics.

The environment is designed with web standards and performance optimization in mind, enabling it to run efficiently in modern browsers without requiring specialized hardware. This accessibility ensures that the system can be deployed across various platforms for both training and operational use.

V. Reinforcement Learning Training Framework

A. Custom OpenAI Gym Environment Implementation

We developed a custom OpenAI Gym environment that formalizes the wildfire mitigation task as a reinforcement learning problem. This environment implements the standard Gym interface while incorporating the complex dynamics of our wildfire simulation:

- Observation Space: The agent receives multi-channel grid observations representing:
  - Fire state (burning intensity across the map)
  - Terrain elevation and gradient
  - Land cover/fuel type classification
  - Wind vector field
  - Agent position and remaining resources
- Action Space: The helitack agent can perform actions including:
  - Movement in cardinal and intermediate directions
  - Water drop operations with variable intensity
  - Hover in place to observe fire behavior
  - Return to base for resource replenishment
- Reward Function: A carefully designed reward structure that balances:
  - Positive rewards for fire suppression (proportional to area saved)
  - Penalties for resource inefficiency (e.g., dropping water on already extinguished areas)
  - Time-dependent penalties that encourage rapid response
  - Terrain-aware bonuses that reward strategic positioning (e.g., targeting fire fronts on steep slopes)
- Episode Termination: Episodes conclude when one of the following occurs:
  - Fire is completely extinguished
  - Fire exceeds containable boundaries
  - Agent depletes all resources
  - Maximum time steps are reached

B. Client-Server Architecture via WebSockets

To enable real-time communication between the Three.js visualization frontend and the Python-based RL backend, we implemented a bidirectional WebSocket communication layer:

- Server Component: A Python server using the websockets library acts as the bridge between the Gym environment and the visualization client. The server:
  - Broadcasts environment state updates to connected clients
  - Receives user interactions or agent actions from clients
  - Synchronizes simulation time steps across components
  - Manages training session state and persistence
- Client Component: The React application integrates WebSocket client functionality to:
  - Render received environment states in the 3D visualization
  - Transmit user commands or agent decisions to the server
  - Display real-time training metrics and agent performance statistics
  - Support observer mode during automated training runs
- Protocol Design: We defined a structured JSON-based message protocol that:
  - Minimizes bandwidth requirements through efficient encoding
  - Supports partial updates to reduce latency
  - Includes sequence numbering for reliable ordered delivery
  - Accommodates both synchronous and asynchronous interaction modes

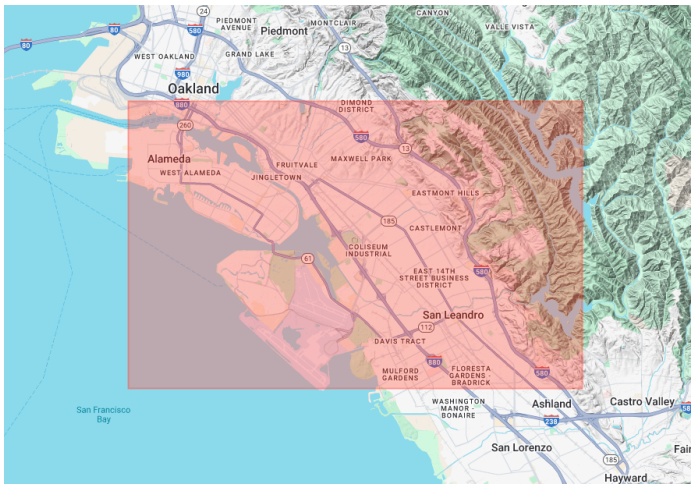


Fig. 3. Terrain visualization in the simulation environment, showing elevation and land cover types.

C. Containerized Training Infrastructure

To ensure reproducibility and scalability of our training process, we implemented a Docker-based training infrastructure:

- Containerization: The entire training stack is encapsulated in Docker containers, including:
  - Python environment with RL libraries and dependencies
  - WebSocket server for communication

  - React client for visualization (headless mode for training)
  - Data processing utilities for geodata preparation
- Training Orchestration: We use Docker Compose to coordinate multiple container instances for:
  - Parallel training of multiple agent variants
  - Hyperparameter optimization runs
  - Distributed evaluation across different wildfire scenarios
  - Performance benchmarking and ablation studies
- Resource Management: The containerized approach enables efficient allocation of computational resources:
  - GPU acceleration for neural network training
  - CPU optimization for simulation physics
  - Memory management for handling multiple concurrent environments
  - Easy deployment across different computing platforms

D. Reinforcement Learning Algorithm Implementation

We implemented and evaluated several reinforcement learning algorithms, ultimately selecting Proximal Policy Optimization (PPO) from the Stable Baselines3 library as our primary approach:

- Policy Network Architecture: Our implementation uses:
  - Convolutional layers to process spatial observation data
  - Attention mechanisms to focus on critical regions (e.g., active fire fronts)
  - Recurrent components (LSTM) to model temporal dependencies in fire behavior
  - Separate value and policy heads for stable learning
- Training Curriculum: To facilitate efficient learning in this complex domain, we developed a progressive curriculum:
  - Initial training on simplified scenarios with single ignition points
  - Gradual introduction of variable wind conditions
  - Progression to complex terrain with diverse land cover
  - Final training on realistic multi-ignition scenarios
- Hyperparameter Optimization: Key parameters were tuned using grid search and Bayesian optimization:
  - Learning rate schedules optimized for stability
  - PPO clip range tuned to prevent policy collapse
  - Entropy coefficient adjusted to balance exploration and exploitation
  - Discount factor (gamma) set to account for delayed effects of firefighting actions

This comprehensive training framework enables our system to learn effective wildfire mitigation strategies that adapt to the complex and dynamic nature of fire behavior across diverse geographical and environmental contexts.

VI. Future Work

While the current stage of the project has successfully established the foundational architecture and demonstrated promising initial results, several key enhancements are planned to extend the system's capabilities, realism, and impact. These future steps are outlined below.

A. Hierarchical Reinforcement Learning

Our current system models a single firefighting strategy (helitack) through a single-agent RL approach. We aim to expand this to a multi-tiered control system using Hierarchical Reinforcement Learning (HRL). In this framework:

- A high-level policy will choose between multiple mitigation strategies, such as deploying fire trucks, helitacks, or constructing firebreaks.
- Low-level policies (sub-agents) will handle the specific execution of each strategy.
- This approach will allow the agent to operate at both strategic and tactical levels, improving overall adaptability and efficiency.

B. Multi-Agent Coordination

In real-world wildfire mitigation, coordination between multiple response units is essential. As a next step, we plan to implement a multi-agent system where:

- Each agent represents a distinct firefighting unit (e.g., ground crew, aerial team).
- Agents may have overlapping but not identical observation spaces and action sets.
- Coordination and communication protocols will be explored using frameworks like MADDPG (Multi-Agent Deep Deterministic Policy Gradient).

C. Policy Transfer to Real Geographies

Currently, our training environments are based on simulated data generated from real-world maps. As the models mature, we intend to:

- Fine-tune trained agents on historical wildfire case studies (e.g., California, Australia).
- Leverage transfer learning techniques to adapt policies across varying terrain and climate conditions.
- Validate agent performance using satellite data and fire event records.

D. Real-Time Decision Support Tool

A major goal of this project is to build a decision-support interface that can assist emergency response teams. Future development includes:

- Deploying the simulation as a cloud-based tool with real-time updates.
- Allowing human operators to interact with the system and adjust RL agent recommendations.
- Incorporating user feedback to dynamically re-train or fine-tune models.

E. Explainability and Trust in RL Decisions

As we deploy RL in high-stakes decision-making scenarios, building trust in the agent's choices is critical. Planned work includes:

- Developing visualization techniques to explain agent decisions.
- Analyzing action trajectories and counterfactuals to understand failure cases.
- Using interpretable RL models or post-hoc explainability tools to aid human oversight.

F. Integration with Government and NGO Systems

Long-term, we envision this system being seamlessly integrated with wildfire management platforms used by:

- Government agencies (e.g., CAL FIRE, NASA FIRMS) for real-time fire spread insights and mitigation planning.
- NGOs involved in environmental protection and disaster response for prioritizing interventions and resource deployment.
- Research organizations focused on climate-resilient infrastructure and AI-powered disaster forecasting.

As a final output, the system generates a concise, human-readable one-page report using a fine-tuned large language model (LLM), summarizing the fire's behavior, estimated spread, and recommended actions. This empowers field officers and policy makers to make fast, informed decisions without needing to interpret complex geospatial data.

By continuing to refine and scale this pipeline, we aim to offer a robust, intelligent, and accessible decision-support framework to help tackle one of the world's most urgent environmental threats.

VII. Bonus: Project Management Tool

For this project, we utilized Trello as our project management tool to ensure structured progress tracking and clear communication. The project was divided into several milestones including data exploration, proposal drafting, basic and advanced model implementations, training, testing, and final presentation preparations.

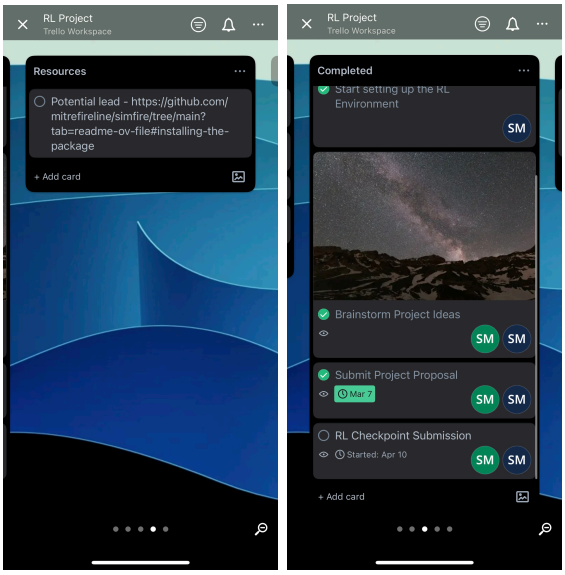
Since our collaboration spanned both the reinforcement learning (RL) and deep learning (DL) parts of the project, both Trello boards were linked. Weekly tasks were clearly documented, enabling effective contribution tracking and synchronization between team members.

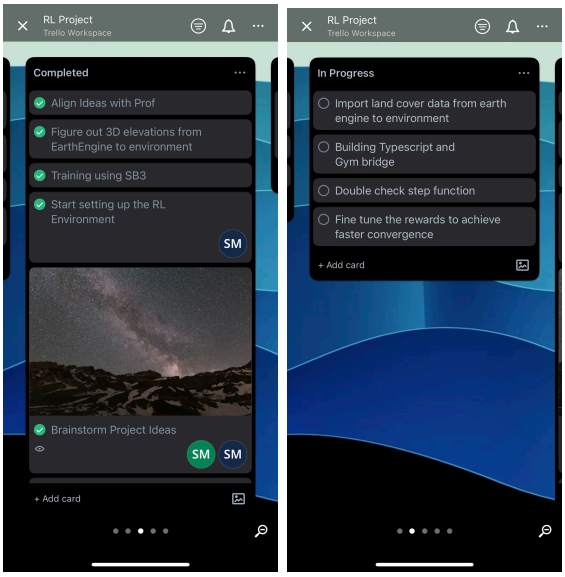
Key milestones included:

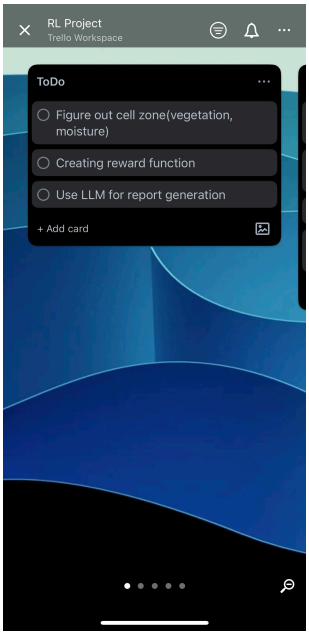
- Project initialization
- Data gathering and preprocessing
- Initial model development
- Advanced model enhancements
- Training and validation of models
- Evaluation and final adjustments

A. Project Management Screenshots

Below are screenshots showcasing our Trello-based project management efforts:







Trello Board URL:
https://trello.com/b/pBNt9VwL/rl-project

GitHub Repository:
https://github.com/ShauryaMathur/inferno-tactix/tree/wildfire-env

B. GitHub Commit History

Below is a snapshot of our git commit history demonstrating ongoing collaboration and iterative development:

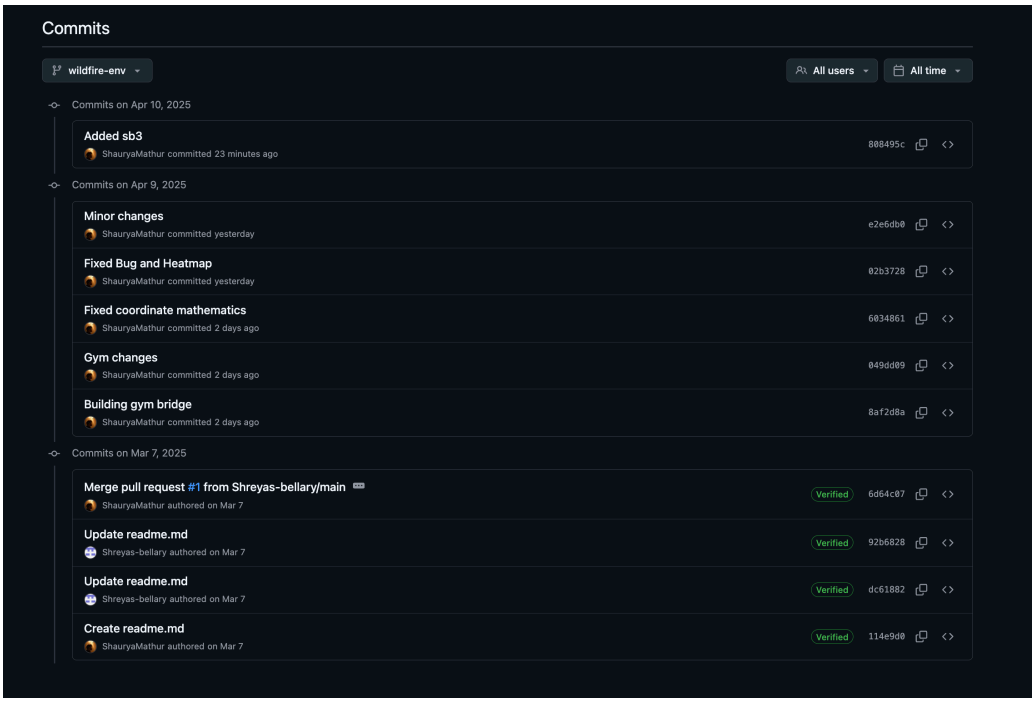


Fig. 4. git commit history

References

[1] https://github.com/amanbasu/wildfire-detection
[2] https://code.earthengine.google.com/
[3] https://wildfire.concord.org/
[4] https://github.com/concord-consortium/wildfire-model/tree/master
[5] U.S. Geological Survey, Landsat 8 (L8) Data Users Handbook, U.S. Geological Survey, 2019. [Online]. Available: https://landsat.usgs.gov/landsat-8
[6] Y. Zhao, M. Chen, and S. Kumar, "A Comprehensive Review of Deep Learning Techniques for Wildfire Detection in Satellite Images," IEEE Access, vol. 9, pp. 10523–10539, 2021.
[7] M. Pereira and G. H. A., "Active Fire Detection in Landsat-8 Imagery: A Large-Scale Dataset," GitHub, 2023. [Online]. Available: https://github.com/pereira-gha/activefire