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—Inferno Tactics presents a novel integrated approach to wildfire management that combines Deep Learning (DL) prediction capabilities with Reinforcement Learning (RL) mitigation strategies. Our system employs a two-phase architecture: first, a deep learning model analyzes historical wildfire patterns and Earth Engine satellite imagery to forecast potential ignition events with precise geographic and temporal specifications. Building upon these predictions, our custom-built 3D simulation environment reconstructs terrain and vegetation conditions from actual geodata, enabling realistic wildfire spread modeling. Within this environment, we've developed and trained a helitack agent using Proximal Policy Optimization (PPO) that demonstrates effective firefighting capabilities through strategic water deployment. The agent successfully learns to prioritize active fire fronts, establish firebreaks in high-risk zones, and efficiently allocate limited resources. Our web application provides an intuitive interface for emergency personnel to input coordinates and receive not only wildfire predictions but also visualized mitigation simulations with actionable strategies. 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. As part of our future work, we plan to implement additional agent types such as ground crews, fire trucks, and fixed-wing aircraft to create a more comprehensive firefighting simulation. Using the Stable Baselines3 library, we train the helitack 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: MODIS land cover dataset, which classifies land into 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 MODIS land cover 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. 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.

Our current implementation uses a grid size of 1200×800 cells, representing a physical area of 120,000×80,000 meters with a maximum elevation of 10,000 meters. For computational efficiency during development and testing, we also maintain a lower-resolution option with a grid width of 240 cells. 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.

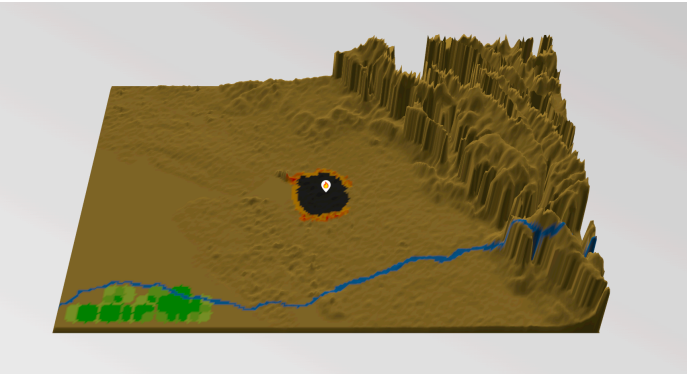


Fig. 2. Interactive 3D wildfire simulation environment developed in Three.js.

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.

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 Time Calculation: Our model computes the ignition probability and timing for each cell using a comprehensive algorithm that integrates multiple environmental variables:
  - Cell moisture content derived from recent precipitation data
  - Vegetation type and density from MODIS land cover classification
  - Terrain elevation and slope gradients affecting fuel preheating
  - Wind speed and direction influencing heat transfer between cells
  - Ambient temperature and relative humidity affecting fuel dryness
- Fire Intensity Dynamics: Each cell's burning intensity is calculated on a continuous scale rather than binary states, allowing for more nuanced fire behavior:
  - Low, medium, and high intensity states determined by fuel load and environmental conditions
  - Variable heat release rates affecting neighboring cell ignition probabilities
  - Intensity-dependent flame height calculations impacting long-distance ember transport
  - Diurnal temperature variations affecting intensity fluctuations throughout simulated time
- Fuel Consumption Model: Each cell maintains a dynamically updated fuel state that evolves as the fire progresses:
  - Differentiated consumption rates based on fuel type (grass burns faster than forest)
  - Moisture-dependent burn efficiency affecting heat output and duration
  - Residual heat calculations that maintain cell temperature after active burning ceases
  - Fuel depletion thresholds that trigger state transitions from burning to burnt
- Strategic Fire Suppression: Our current implementation focuses exclusively on helitack operations:
  - Water drops create cooling zones with radius proportional to drop intensity
  - Suppression effectiveness varies based on fire intensity and fuel characteristics
  - Strategic placement is critical with limited water resources and refill requirements

- Our RL agent learns to prioritize containment lines and high-risk spread vectors rather than targeting the most intense portions of the fire

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 Gymnasium interface while incorporating the complex dynamics of our wildfire simulation through a WebSocket connection to our React-based frontend:

- Observation Space: The agent receives multi-dimensional observations structured as a dictionary:
  - helicopter_coord: Box space containing the agent's x,y position (range 0-239, 0-159)
  - cells: A 4-channel grid (160×240) representing stacked frames of the environment state, where each cell's value encodes both fire state (unburnt, burning, burnt) and intensity level (low, medium, high)
  - on_fire: Binary indicator of whether any cell in the environment is currently burning
- Action Space: The helitack agent can perform five discrete actions:
  - Move in four cardinal directions at a fixed speed (HELICOPTER_SPEED=3)
  - Perform a water drop operation at the current location
- Reward Function: Our reward mechanism incorporates multiple components to guide the agent toward effective firefighting strategies:
  - +10 reward for each cell directly extinguished by helitack action
  - -5 penalty for each newly burnt cell (discouraging fire spread)
  - -0.1 penalty per burning cell (proportional to current fire size)
  - -0.1 time penalty to encourage efficient action
  - Proximity rewards based on distance to nearest fire ($2 \times e^{-\text{distance}/20}$)
  - Intensity-based bonuses for targeting high-intensity fires (up to 3× multiplier)
  - Strategic firebreak rewards for preventive water drops near burning areas
  - Penalties for wasteful drops on already burnt (-5) or far-from-fire unburnt areas (-10)
- Frame Stacking: To enable the agent to perceive fire dynamics and spread patterns:
  - Environment maintains a history of 4 consecutive frames
  - New observations replace the oldest frame using an efficient rolling algorithm
  - This temporal context allows the agent to infer fire movement direction and speed
- Episode Termination: Episodes conclude when either:
  - Fire is completely extinguished (cellsBurning = 0)
  - Maximum time steps are reached (MAX_TIMESTEPS = 2000)

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

C. Neural Architecture for Fire Environment Feature Extraction

A critical component of our reinforcement learning framework is the custom feature extraction architecture designed specifically for wildfire environments. The spatial complexity and dynamic temporal patterns of fire spread necessitate a specialized neural network that can effectively process multi-scale patterns and temporal sequences. To address this challenge, we developed a multi-faceted feature extraction pipeline:

1) Multi-Scale Convolutional Analysis: Our feature extractor employs a parallel multi-scale CNN approach to simultaneously capture fire dynamics at different spatial resolutions:

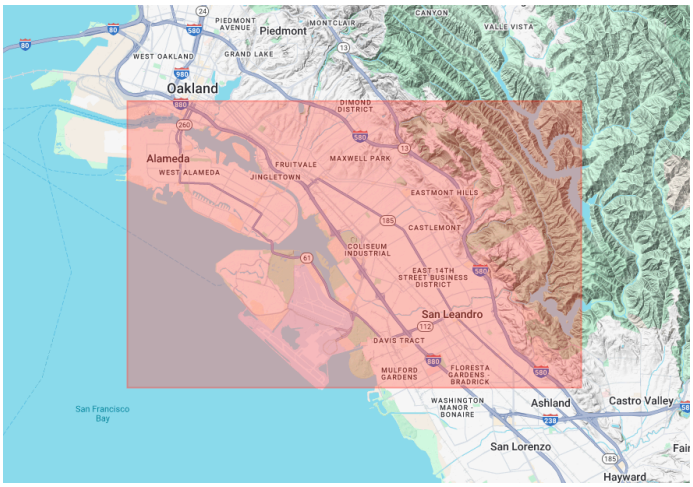


Fig. 3. Terrain visualization in the simulation environment, showing elevation and land cover types.

- Small Receptive Field Network: Processes local fire details with 3×3 kernels, capturing fine-grained burning patterns and precise boundaries between fire states.
- Medium Receptive Field Network: Identifies fire clusters and potential spread pathways using 5×5 kernels, enabling the agent to recognize emerging fire fronts and strategic intervention points.
- Large Receptive Field Network: Provides global context with 11×11 and 7×7 kernels, allowing the agent to understand overall fire distribution and macro-level patterns across the terrain.

This multi-resolution approach ensures that the agent simultaneously perceives both localized details (which cell is burning at what intensity) and global patterns (overall fire shape and spread direction), mirroring how human firefighters assess wildfire scenarios across different scales.

2) Spatial Attention Mechanism: To focus computational resources on the most relevant regions of the environment, we implemented a spatial attention mechanism that enhances important features:

- The attention module generates a pixel-wise weighting map that emphasizes critical areas such as active fire fronts and potential spread vectors.
- A residual connection (attended features + original features) ensures that no information is lost during the attention process while still highlighting priority regions.
- This mechanism enables the agent to prioritize actions in high-risk areas without completely ignoring secondary concerns, much like how human firefighters must allocate limited resources to the most critical fire zones.

3) Temporal Understanding via LSTM: Fire behavior is inherently temporal, with spread patterns evolving over time based on terrain, wind, and intervention effects. To capture these dynamics:

- We incorporate a two-layer LSTM network with 256 hidden units that processes the sequence of observations, enabling the agent to recognize temporal patterns in fire spread.
- The LSTM maintains internal state between time steps, allowing the agent to develop an understanding of fire momentum, spread rates, and the effectiveness of previous interventions.
- This recurrent architecture helps the agent anticipate future fire movement rather than simply reacting to the current state, enabling more strategic planning and proactive firefighting.

4) Coordinate and Fire State Integration: To complete the agent's situational awareness, we process additional contextual information:

- The helicopter's absolute and relative position (distance from center) is encoded through a specialized coordinate network, helping the agent develop an understanding of spatial relationships and egocentric navigation.
- A binary fire status flag is processed to provide immediate feedback on whether any cells are currently burning, serving as a high-level indicator of mission status.
- These features are concatenated with the CNN and LSTM outputs to create a comprehensive representation of the environment state.

5) Integration with PPO Algorithm: The feature extractor is integrated into our PPO implementation through a custom policy class that connects the extractor to the actor and critic networks:

- The actor network maps the extracted features to a probability distribution over the five possible actions (four movement directions and water drop).
- The critic network estimates the value function from the same features, enabling effective advantage estimation for policy updates.
- During training, the entire architecture—including the feature extractor—is optimized end-to-end, allowing the perception components to adapt specifically to the wildfire suppression task.

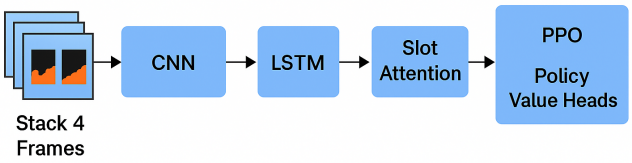


Fig. 4. Architecture of our FireEnvLSTMCNN feature extractor with multi-scale convolutional pathways, spatial attention, and LSTM temporal processing.

D. 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

E. 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 helitack agent to learn effective wildfire mitigation strategies that adapt to the complex and dynamic nature of fire behavior across diverse geographical and environmental contexts. In future iterations, we plan to extend this framework to support multiple agent types that can operate collaboratively.

VI. Interactive Web Application

To make our wildfire prediction and mitigation system accessible to a wider audience, we have developed a user-friendly web application that facilitates interaction with our trained models. This interface serves as a bridge between the technical complexity of the AI models and the practical needs of emergency management personnel.

A. User Interface and Workflow

The web application provides an intuitive interface where users can:

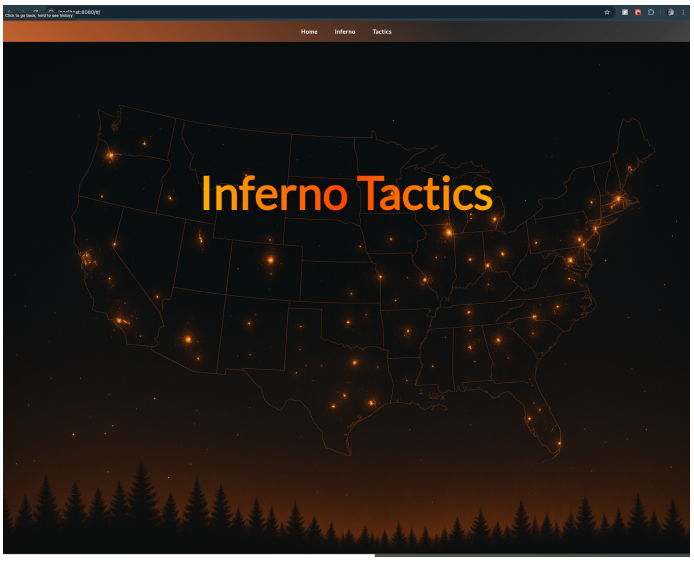


Fig. 5. Web Application Home Page

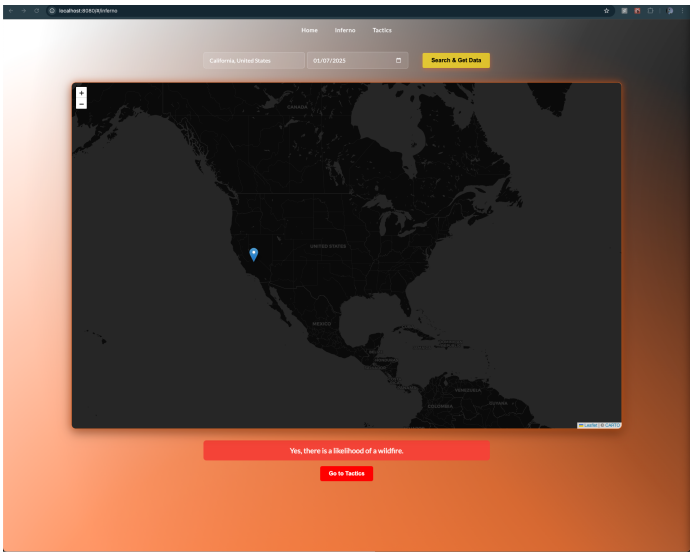


Fig. 6. Inferno Prediction Page

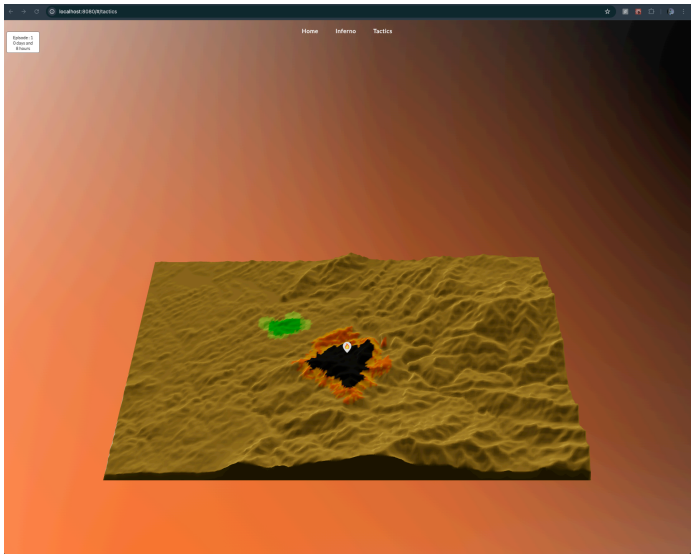


Fig. 7. Tactics Mitigation Page

- Input Location: Specify geographic coordinates (latitude and longitude) or use an interactive map to select a region of interest.
- Select Date Range: Choose a future date or date range for wildfire risk assessment.
- View Prediction Results: Receive a binary prediction (fire/no fire) along with a confidence score indicating the likelihood of wildfire occurrence.

If a wildfire is predicted with high confidence, the application automatically triggers the following workflow:

1) Data Acquisition: The system fetches elevation data from SRTM and land cover data from MODIS for the specified location.
2) Environment Generation: Using the acquired geodata, the application dynamically creates a simulation environment with dimensions of 1200×800 grid cells (representing a physical area of 120,000×80,000 meters with a heightmap maximum elevation of 10,000 meters).
3) Mitigation Simulation: The pre-trained reinforcement learning agent is deployed in this environment to demonstrate optimal fire suppression strategies.
4) Interactive Exploration: Users can observe the agent's recommended actions, adjust parameters such as wind conditions or resource availability, and explore alternative scenarios.

B. Technical Implementation

The web application is built using modern web technologies:

- Frontend: React.js with Three.js for 3D visualization
- Backend: Flask server that handles prediction requests and coordinates with the Google Earth Engine for geodata retrieval

- Model Deployment: TensorFlow Serving for the deep learning prediction model and a custom server for the reinforcement learning policies
- Data Pipeline: Automated workflow that processes user inputs, generates predictions, and prepares simulation environments

The integration with our simulation environment allows for seamless transition from prediction to action planning. Once the environment is loaded with the relevant geographic data, users can visualize potential fire spread patterns and explore different mitigation approaches before committing resources in the real world.

C. Practical Applications

This web application transforms our research into a practical tool for various stakeholders:

- Emergency Management Teams: Can use the platform for proactive planning and resource allocation based on predicted fire risks.
- Fire Departments: May leverage the simulation capabilities to train personnel on optimal response strategies for specific geographic contexts.
- Land Management Agencies: Can identify high-risk areas and implement preventive measures such as controlled burns or firebreak construction.
- Research Community: Can utilize the platform to test new wildfire models or adaptation strategies in a realistic virtual environment.

By providing this accessible interface to our AI-driven wildfire prediction and mitigation system, we aim to bridge the gap between advanced research and practical application in the field of disaster management.

VII. 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. Meta-Reinforcement Learning for Adaptation

A crucial enhancement to our system will be the integration of Meta-Reinforcement Learning (Meta-RL) capabilities. Wildfire environments are characterized by constant change—variations in weather patterns, fuel conditions, terrain types, and fire behavior make each wildfire event unique. Traditional RL agents often struggle to generalize when faced with environments that differ significantly from their training distribution.

- Rapid Adaptation: Meta-RL frameworks will enable our agents to quickly adapt to new, previously unseen environmental conditions without extensive retraining. This is particularly valuable for wildfire management, where conditions can change rapidly during a single incident.
- Transfer Learning Across Environments: By framing wildfire mitigation as a meta-learning problem, we can train agents that learn "how to learn" optimal strategies for different fire scenarios, rather than learning fixed policies for specific conditions.
- Context-Aware Decision Making: Meta-RL algorithms like Model-Agnostic Meta-Learning (MAML) and Reptile will allow our agents to infer the current environmental dynamics from limited observations and adapt their strategies accordingly.
- Continual Improvement: As the system encounters more wildfire scenarios, the meta-learning component will progressively improve its adaptation capabilities, making the entire system more robust to environmental variability.

The implementation of Meta-RL will be particularly beneficial when expanding beyond our current single helitack agent to include multiple agent types. Each agent class can leverage meta-learning to specialize in certain environmental contexts while maintaining adaptability across a range of conditions.

B. 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.

C. 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 that extends beyond our current single helitack agent:

- Each agent represents a distinct firefighting unit (e.g., ground crew, fire truck, fixed-wing aircraft, drone reconnaissance).
- 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).

D. 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.

E. 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.

F. 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.

G. 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.

VIII. 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.

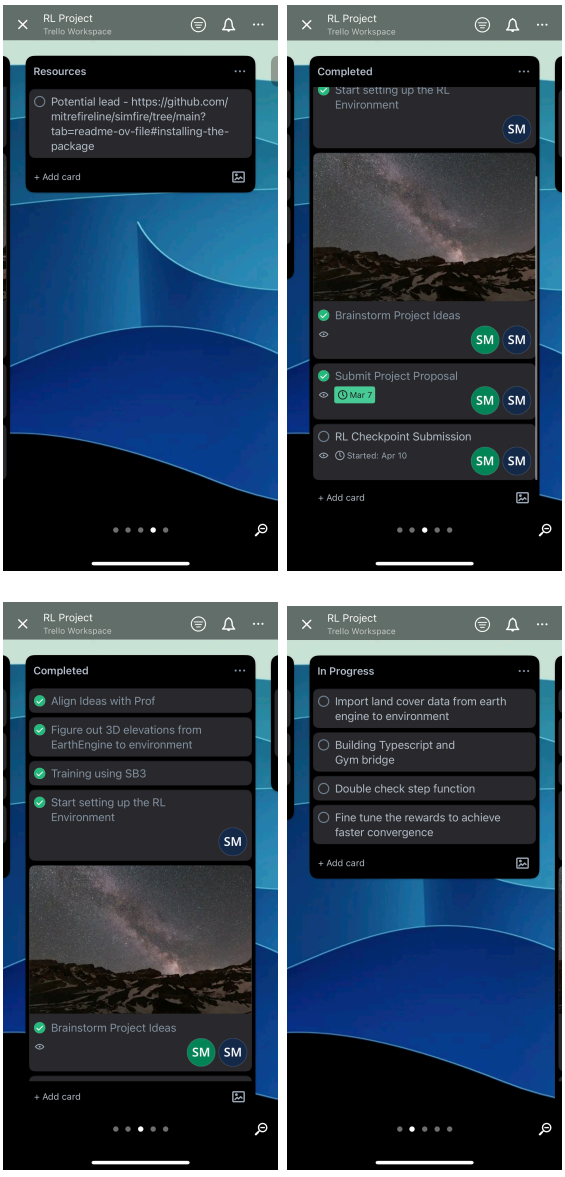
Since our collaboration spanned both the reinforcement learning (RL) and deep learning (DL) parts of the project, both Trello boards were linked. Weekly tasks were clearly documented, enabling effective contribution tracking and synchronization between team members.

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:



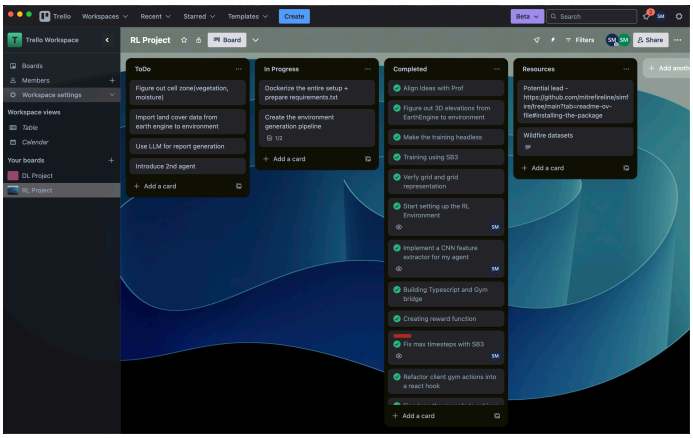


Fig. 8. Trello Dashboard

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:

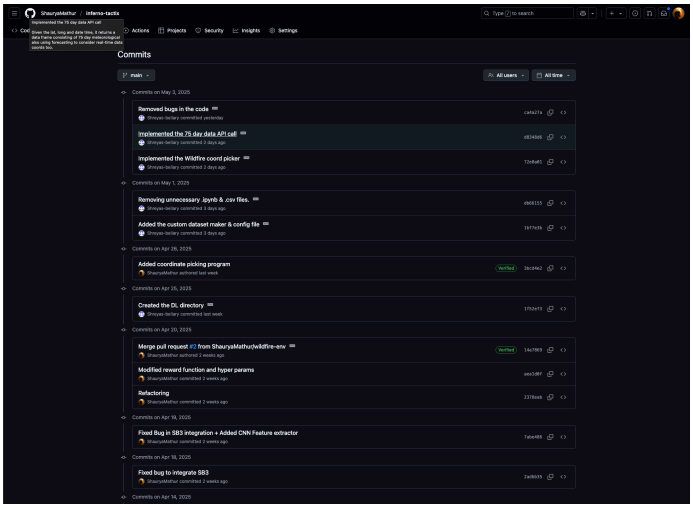


Fig. 9. Git commit history

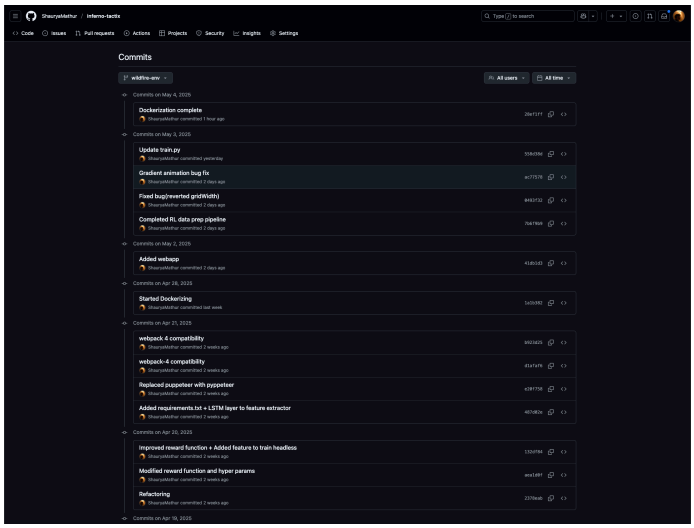


Fig. 10. Git commit history

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
[8] https://earthengine.google.com/
[9] https://medium.com/@AlexanderObregon/building-real-time-applications-with-python-and-websockets-eb33a4098e02
[10] https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html