Predicting Movement Paths in Search and Rescue Operations Using RL

Junhee Kim, Heedam Kwon, Seongil Jo, Jaeoh Kim
Inha University

인하대학교

INHA UNIVERSITY

Abstract

In search and rescue operations, achieving operational success is challenging even when substantial resources are deployed. This study develops a predictive model for missing person movement paths using reinforcement learning (RL) techniques based on deep neural networks.

The research compares Vanilla Deep Q-Network (DQN) and **Pointer Network DQN**, demonstrating that the Pointer Network DQN achieves more accurate and consistent path predictions in complex terrain conditions.

Data

1. Geospatial Data:

- **Digital Elevation Model (DEM):** Represents terrain height.
- **Shapefiles:** Includes forest roads, rivers, hiking trails, and accident points.

2. Data Preprocessing:

- Rasterization of geospatial data to grid format.
- Binaryization to express the presence or absence of a particular terrain
- Calculate the DEM to extract additional topographic features such as slope, roughness, etc

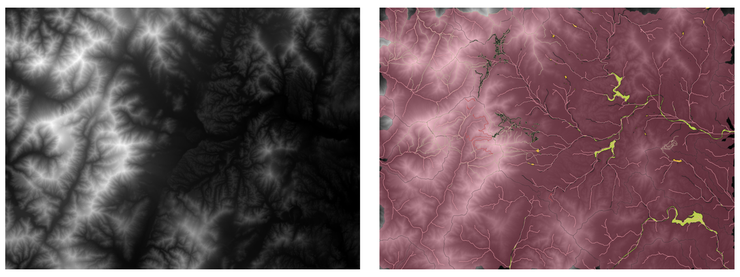

Figure 1: Geospatial Data Visualization: DEM and Shapefiles Example

Model Architectures

1. Vanilla DQN:

- Simple feedforward neural network with two hidden layers.
- Outputs Q-values for eight possible movement directions.

2. Pointer Network DQN:

- Encoder-decoder LSTM architecture with attention mechanism.
- Designed to handle complex terrain features and predict optimal paths.

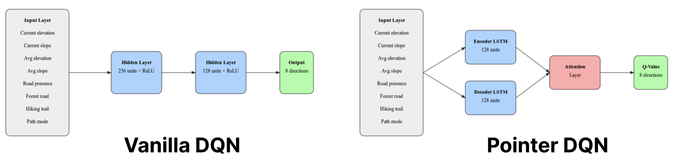

Figure 2: Vanilla DQN & Pointer DQN Architecture

Model Inputs:

- The state is represented as an 8-dimensional feature vector:
  - `self.dem[y, x]`: Elevation at the current position.
  - `self.slope[y, x]`: Slope at the current position.
  - `np.mean(dem_patch)`: Average elevation in the surrounding patch.
  - `np.mean(slope_patch)`: Average slope in the surrounding patch.
  - `np.max(road_patch)`: Maximum road presence indicator in the patch.
  - `np.max(forestroad_patch)`: Maximum forest road presence indicator in the patch.
  - `np.max(climbpath_patch)`: Maximum climbing path presence indicator in the patch.
  - `np.std(dem_patch)`: Standard deviation of elevation values in the patch.

Actions(Model Output):

- 8 probabilities for 8 actions in a particular state are returned.

Reward Function Design

Designing the Reward Function for Optimal Pathfinding:

- Rewards are crafted to incentivize optimal behaviors and penalize inefficiencies.
- Each component of the reward function is calibrated for effective learning in complex terrains.

Reward/Penalty	Description
Path following reward	Base reward (+0.5) for staying on the predefined path.
Water proximity reward	Additional reward (+0.15) for being near water without entering it.
Directional consistency	Reward (+0.1) for maintaining the same direction as previous movement.
Exploration reward	Encourages new locations by adding a small reward (+0.02) for not revisiting previous positions.
Diversity penalty	Penalty (-0.1) for repetitive back-and-forth movements within a small area.
Stationary penalty	Penalty (-0.2) for not moving from the current position.
Revisit penalty	Penalty proportional to revisit count (-0.05 × number of visits) for the same position.
Slope penalty	Penalty (-0.05) for traversing areas with a slope greater than 30 degrees.
Water penalty	Large penalty (-0.5) for entering water regions.
Boundary reward	Bonus reward (+0.03) for staying within the predefined search area.
Proximity reward	Reward (+0.3 / distance) for moving closer to the predefined path during exploration mode.

Training Process

1. Data Preparation

- Processing DEM and shapefile data through rasterization and binarization
- Converting environmental data into 8D state vectors containing elevation, slope, trail presence, water proximity, and terrain features
- Normalizing all features to [0,1] range for consistent network input

2. Training Setup

- Experience buffer size: 10,000 samples
- Epsilon-greedy exploration: 0.99 $\rightarrow$ 0.1 with scheduler
- Network updates: Adam optimizer (lr = 0.001)
- Target network synchronization: Every 1000 steps
- Maximum steps per episode: 1000
- Discount factor (γ): 0.99 for future reward consideration

3. Model Checkpointing

- Best model preservation based on reward metrics
- Regular checkpoints every 10 episodes
- Early stopping when performance plateaus
- Training statistics logged for performance analysis

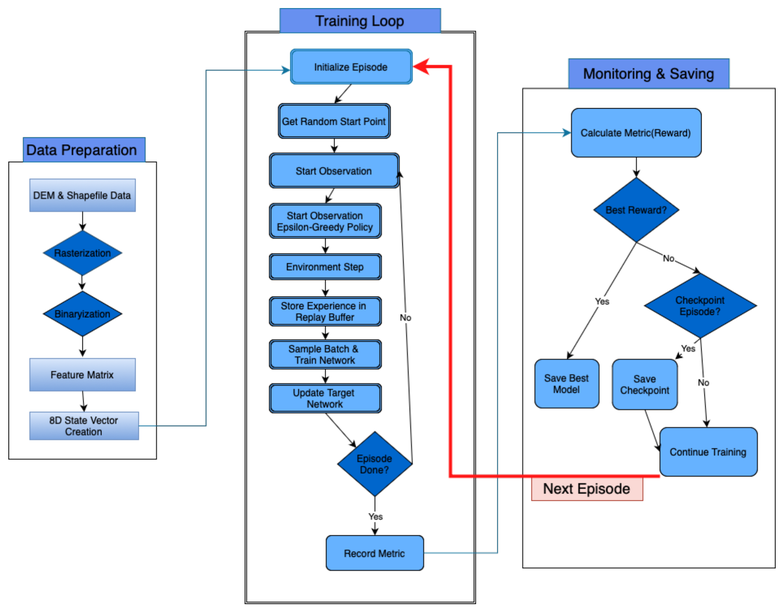

Figure 6: Training process workflow showing data preparation, training loop, and monitoring stages

Results

Performance Metrics

- Vanilla DQN:
  - Average Reward: +8.8 ± 1.2
  - Success Rate(Find Road Rate): 81.3%
- Pointer Network DQN:
  - Average Reward: +10.2 ± 0.8
  - Success Rate(Find Road Rate): 94.7%

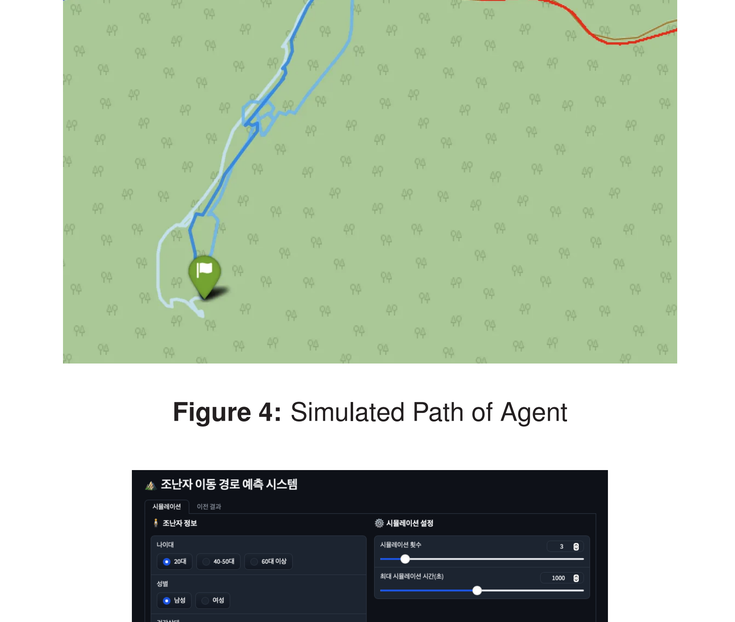

Figure 3: Average Reward & Loss Comparison

Figure 4: Simulated Path of Agent

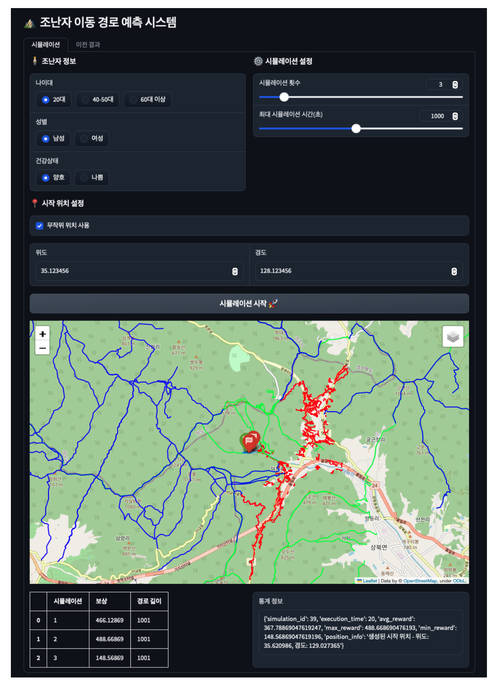

Figure 5: Gradio Interface for Displaying Simulation Path

Conclusion & Future Work

The Pointer Network DQN demonstrates significant improvements compared to the Vanilla DQN in terms of reward and loss, achieving much faster convergence. Additionally, it exhibited the ability to understand and adapt to complex and diverse terrain relationships through the Attention Mechanism.

Although the model considered various types of terrain, it was limited by the sparsity of geographic data such as water-related features and trail information.

For future research, alternative reinforcement learning methodologies such as Proximal Policy Optimization (PPO) or Soft Actor-Critic (SAC) could be explored. These methods offer potential advantages in terms of stability and sample efficiency. PPO's conservative policy updates could help maintain consistent performance across different terrain types, while SAC's entropy maximization approach might lead to more robust exploration strategies in complex environments.