International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Developing Self-Healing and Self-Pptimizing Networks Using Deep Reinforcement Learning

Dhase Ganesh Narsinhrao[1], Deshmukh Jaideep V[2], Swami G D[3], Gajanan K Damkondwar[4]

[1]H.O.D., C M Department, MSPMs Yashwantrao Chavan Institute of Polytechnic Beed,
[2]H.O.D., AIML Department, MSPMs Yashwantrao Chavan Institute of Polytechnic Beed,
[3,4]Lecturer, C M Department, MSPMs Yashwantrao Chavan Institute of Polytechnic Beed,

Abstract
Next-generation networks (such as 5G/6G and edge computing) require unprecedented levels of reliability and efficiency, rendering manual or heuristic network management obsolete. To achieve true autonomy, networks must possess self-healing and self-optimizing capabilities. Deep Reinforcement Learning (DRL) offers a powerful framework for this, but standard DRL methods often suffer from slow convergence, scalability issues, and policy instability when handling dual-objective tasks in large-scale topologies. In this paper, we propose [Insert Name of Your Framework], an intelligent, self-configuring networking framework driven by an advanced DRL architecture. Our approach models localized network telemetry as a Markov Decision Process (MDP) to dynamically detect structural disruptions and optimize resource allocation simultaneously. By incorporating [insert key algorithmic novelty, e.g., a dual-attention mechanism / localized reward structuring], the proposed framework accelerates agent convergence and avoids catastrophic policy shifts during sudden link failures. We evaluate our method across various standard network topologies under volatile traffic demands. Simulation results demonstrate that our framework achieves a [X%] reduction in fault recovery latency and a [Y%] improvement in overall throughput compared to state-of-the-art baselines, paving the way for resilient, zero-touch network automation

Keywords: Deep Reinforcement Learning (DRL), self-healing networks, self-optimizing networks, zero-touch network management, Markov Decision Process (MDP), 5G/6G communication networks, network resilience, quality of service (QoS), autonomous traffic engineering.

Introduction

The unprecedented explosion of heterogeneous traffic generated by next-generation technologies—such as 5G/6G communication, internet of things (IoT), and edge computing—has fundamentally challenged conventional network architectures. Modern communication networks are characterized by highly dynamic topologies, volatile traffic demands, and strict quality of service (QoS) constraints. Managing these environments using manual intervention or static, rule-based systems is no longer viable due to human latency and the inability to foresee stochastic network failures. Consequently, the paradigm shift toward cognitive, autonomic networking—specifically focusing on self-healing and self-optimizing functionalities—has become paramount for ensuring operational resilience and service continuity.

By representing complex network infrastructures as Markov Decision Processes (MDPs) as shown in the control loop above, DRL enables autonomous agents to interact dynamically with the environment. The agent continuously observes state characteristics s_t (e.g., throughput, buffer occupancy, latency) from the network environment, executes optimization actions a_t, and receives a scalar reward r_t designed to maximize long-term operational efficiency. This experience-driven trial and error allows the system to learn optimal control policies without requiring explicit mathematical models of the traffic dynamics.

The concepts of self-healing and self-optimization dictate that a network must autonomously discover, diagnose, and remediate faults (such as link failures or node outages) while simultaneously adjusting routing strategies and resource allocation to optimize performance parameters. While early automation relied on heuristic thresholding, these approaches struggle in massive, unpredictable environments. To bridge this gap, Deep Reinforcement Learning (DRL) has emerged as a disruptive framework.

Despite its promise, implementing DRL for dual self-healing and self-optimizing tasks introduces severe challenges: Action-Space Explosion: As network topologies scale, the combinatorial explosion of potential routing paths and bandwidth allocation choices creates massive action spaces that make policy exploration mathematically intractable.

Conflicting Optimization Objectives: Maximizing steady-state performance metrics (like total throughput) often directly conflicts with self-healing protocols, which require preserving backup capacity for sudden topological faults.

Convergence Latency: Standard deep reinforcement algorithms require millions of training iterations to stabilize, rendering them dangerously slow when reacting to real-time physical link degradation.

RELATED WORK

The table below provides a structural comparison of recent state-of-the-art Deep Reinforcement Learning (DRL) and intelligent networking architectures. It maps their specific focus areas (Self-Healing vs. Self-Optimizing), core algorithms, performance metrics, and primary technical limitations.

Research / Framework	Core Focus Area	DRL / Optimization Algorithm	Target Network Environment	Key Performance Metrics	Primary Technical Limitations
DRL-SHF (Menaka, n.d.)	Self-Healing (Primary focus on dynamic topological fault recovery)	Proximal Policy Optimization (PPO) with Generalized Advantage Estimation (GAE)	Heterogeneous 5G, Multi-access Edge Computing (MEC), and autonomous IoT meshes	32.6% reduction in packet loss 27.8% latency improvement over standard RL	High structural variance during massive concurrent node drops. Relies heavily on accurate

Research / Framework	Core Focus Area	DRL / Optimization Algorithm	Target Network Environment	Key Performance Metrics	Primary Technical Limitations
					proactive fault prediction modeling.
TS2RLA Framework (Fatima, n.d.)	Self-Healing Self-Optimizing (Joint optimization of link integrity & throughput)	Attention-Based Deep Reinforcement Learning	Mixed-autonomy environments & variable-dimension multi-factor topologies	86.74% reduction in systemic crashes 71% improvement in operational throughput	Scalability is bottlenecked when sequence input sizes scale rapidly. Suffers from compute overhead due to multi-head attention over large topologies.
Continuous Actuation Model (Chatterjee et al., 2025)	Self-Healing	Twin Delayed Deep Deterministic Policy Gradient (TD3) vs. DQN/Q-Learning	Stochastic simulation environments with vascular/mesh repair dynamics	Restored structural integrity to ≈ 1.0 within 5–6 steps Drastically optimized resource efficiency	Continuous action tracking leads to slow exploration phases in high dimensions. Performanc

Research / Framework	Core Focus Area	DRL / Optimization Algorithm	Target Network Environment	Key Performance Metrics	Primary Technical Limitations
					e is highly sensitive to the design of the reward penalty coefficients .
Intelligent WSN Framework (Alauthman et al., 2025)	Self-Healing Self-Optimizing (Joint fault detection & traffic reconfiguration)	LightGBM (Fault Detection) + Flying Fox Optimization Algorithm (FFOA)	Dynamic Wireless Sensor Networks (WSNs)	94.6% fault detection accuracy 120 ms minimum recovery time 98.5% network resilience	Hybrid mapping relies on two separate execution blocks, increasing edge deployment footprint. FFOA heuristic can stall in local optima under explosive traffic surges.
DRL-NSABC Baseline (Amirkhanova et al., 2026)	□ Self-Optimizing (Anomalous pattern detection & adaptation)	Hybrid CNN-RNN driven DRL optimized via Meta-Heuristics	Industrial Smart Metering & Cyber-Physical Data Layers	• Accurate adaptive feature parsing in complex correlated d	

Python Code Template: Self-Healing Environment

This script creates a custom environment based on Open AI Gym or Farama-Foundation Gymnasium, representing a network with a hidden fault.

```
import numpy as np
import gymnasium as gym
from gymnasium import spaces

class SelfHealingNetworkEnv(gym.Env):
"""
A simulated mesh network environment for testing DRL self-healing.

Goal: Bypassing faulty/anomalous paths to restore link quality.

"""
def __init__(self):
super(SelfHealingNetworkEnv, self).
__init__()

# Action Space: 4 discrete alternative routing paths (Path 0 to Path 3)
self.action_space = spaces.Discrete(4)

# State Space: [Path 0 Quality, Path 1 Quality, Path 2 Quality, Path 3 Quality]
# Values range from 0.0 (Dead/Anomalous) to 1.0 (Optimal performance)
self.observation_space = spaces.Box(low=0.0, high=1.0, shape=(4,), dtype=np.float32)

# Internal configuration
self.state = np.ones(4, dtype=np.float32)
self.faulty_path = None
self.steps = 0

def reset(self, seed=None, options=None):
super().
reset(seed=seed)
self.steps = 0

# Initialize all network paths to an optimal state
self.state = np.ones(4, dtype=np.float32)

# Inject an unexpected random node/soft-fault into one of the paths
self.faulty_path = np.random.randint(0, 4)
self.state[self.faulty_path] = np.random.uniform(0.0, 0.2) # Severe degradation
```

```
return self.state, {}

def step(self, action):
self.steps += 1

# Evaluate agent's routing decision
chosen_path_quality = self.state[action]

# Calculate Reward Function:
# High reward for routing through healthy paths; heavy penalty for hit paths
if chosen_path_quality &gt; 0.7:
reward = 10.0  # Successful bypass/healing action
else:
reward = -20.0 # Failed action: Routed traffic into the fault zone

# Dynamically shift soft-faults over time to simulate a real network
if self.steps % 5 == 0:
self.state = np.ones(4, dtype=np.float32)
self.faulty_path = np.random.randint(0, 4)
self.state[self.faulty_path] = np.random.uniform(0.0, 0.2)

# Terminate episode after 20 network adjustments
terminated = self.steps &gt;= 20
truncated = False

return self.state, reward, terminated, truncated, {}
```

Designing the Reward Function for Fault Isolation

To stop the agent from constantly switching paths (thrashing), use a complex reward function that includes a buffer for stability:

reward = (alpha * path_throughput) - (beta * packet_drop_rate) - (gamma * routing_switch_overhead)

• Routing Switch Overhead: Adds a small penalty each time the agent

Example of List of References

1. Fatima, S. (n.d.). Self Healing of a Mixed Autonomy Traffic System Using Reinforcement Learning and Attention. *IEEE Xplore*. Cited by: 2
2. Menaka, G. (n.d.). Deep Reinforcement Learning for Self-Healing Communication Networks: Addressing Node Failure and QoS Degradation in Dynamic Topologies. *National Journal of Antennas and Propagation*, 134-142. Cited by: 1