Chaos Engineering for AI Services: A Production-Safe Framework for Multi-Provider Reliability Testing

|  |  |  |
| --- | --- | --- |
| Yashveer Ahlawat  yashveer.ahlawat.22cse@bmuedu.in  BML Munjal University  Gurgaon, Haryana, India | Gautam Shukla  gautam.shukla.22cse@bmu.edu.in  BML Munjal University  Gurgaon, Haryana, India | Gaurav Chaudhary  gaurav.chaudhary.22cse@bmu.edu.in  BML Munjal University  Gurgaon, Haryana, India |

# ABSTRACT

# Service like ChatGPT, Google AI, and Claude are now essential parts of modern applications. However, making sure these AI services work reliably is a big challenge. Traditional chaos engineering (a method used by Netflix to test system reliability) works well for regular computer systems, but it doesn't handle the unique problems that AI services face. These problems include API rate limits, login failures, incomplete responses, and managing multiple AI providers at once.

# This paper presents the AI Resilience Monitor, a complete system that takes Netflix's proven chaos engineering method and adapts it specifically for AI services. Our system includes nine types of AI-specific failure tests, smart routing between multiple AI providers with circuit breakers, and safe testing methods designed for commercial AI environments. We use the four main principles of chaos engineering - defining normal behaviour, simulating real-world problems, testing in production safely, and automating everything - to show how traditional reliability methods can work for AI services.

# Our system also includes a new scoring method that measures how resilient AI services are across different areas, giving clear numbers to show how ready they are for production use. This work fills the important gap between existing chaos engineering practices and what AI services actually need. It provides both research contributions and practical tools for companies using AI services. This is the first complete application of chaos engineering designed specifically for AI service reliability, creating a foundation for future research in this growing field.

# KEYWORDS

# INTRODUCTION

The landscape of modern computing has been fundamentally transformed by the widespread adoption of artificial intelligence services. From conversational AI systems like ChatGPT and Google Gemini to specialized AI APIs for computer vision, natural language processing, and predictive analytics, AI services have become integral components of contemporary software architectures. This proliferation has created unprecedented dependencies on third-party AI providers, introducing novel failure modes and reliability challenges that existing testing methodologies fail to address comprehensively.

1.1 Problem Definition

The central problem addressed by this research is the absence of systematic methodologies for testing AI service reliability under failure conditions. Organizations deploying AI-dependent applications face significant risks from AI service outages, authentication failures, rate limiting events, and provider-specific disruptions, yet lack scientific frameworks for measuring and improving resilience against these failure modes.

Existing chaos engineering tools focus primarily on infrastructure-level failures such as server termination, network partitioning, and resource exhaustion. While these remain relevant, they fail to address AI-specific failure scenarios including API authentication failures, quota exhaustion events, model unavailability situations, provider-specific error patterns, and multi-provider coordination challenges. This gap leaves organizations vulnerable to AI service disruptions that could have been anticipated and mitigated through appropriate resilience testing.

1.2 Project Objectives

This project establishes clear objectives addressing the identified research gap. The primary objective is to develop the first comprehensive chaos engineering framework specifically designed for AI service reliability testing. This encompasses implementing all four chaos engineering principles as defined by Basiri et al., adapted for AI service contexts, creating comprehensive AI-specific failure injection capabilities covering authentication, rate limiting, partial response, and provider availability scenarios, and designing production-safe experiment frameworks enabling controlled resilience testing.

Secondary objectives include developing real-time monitoring and assessment capabilities through interactive dashboards and automated readiness scoring, implementing multi-provider routing and failover mechanisms optimized for AI service characteristics, creating scientific validation frameworks for measuring resilience improvements, and establishing extensible architectures supporting additional AI service types and failure modes. The ultimate goal is demonstrating practical applicability through comprehensive system implementation and validation, establishing a foundation for future research in AI service reliability engineering.

Through achieving these objectives, this project addresses a critical need in modern software engineering while advancing the theoretical foundations of chaos engineering into new application domains. The resulting system provides organizations with practical tools for improving AI service reliability while contributing to academic understanding of resilience testing methodologies for emerging technologies.

# MOTIVATION

**2.1 Why AI Service Reliability Matters Now**

**Real Business Impact**: Specific examples like Amazon's 20-30% revenue from AI recommendations

**Critical Scenarios**: Healthcare diagnostics, financial trading, emergency response systems

**Soft Failures**: Explained how AI services degrade gradually rather than failing completely

**External Dependencies**: Why you can't directly fix AI provider problems.

**2.2 Problems with Traditional Methods**

**Netflix Example**: Concrete explanation of how Chaos Monkey works and why it doesn't fit AI services

**Specific Testing Scenarios**: Rate limiting, API key expiration, cached responses, backup provider differences

**Economic Constraints**: Why traditional high-volume testing is financially impractical

**Authentication**: API key expiration scenarios, backup credential management

**Partial Responses**: Code completion AI switching to simpler models under load

**Multi-Provider**: OpenAI + Claude + Gemini routing complexity with context preservation

**Quality Degradation**: Translation accuracy, summarization detail, confidence levels

**Provider-Specific Errors**: Different error handling across AI providers

# 3 METHODOLOGY

# 4 CHARACTERIZATION AND ANALYSIS

# 5 TOWARDS CARBON-EFFICIENT ML SERVING

# 6 SUMMARY AND FUTURE DIRECTIONS

# ACKNOWLEDGMENTS

# REFERENCES