Production-Ready MLOps for Telecommunications: Neural Network-Based VoIP Monitoring with 88% Accuracy and Automated Deployment Pipelines

Transforming network operations through AI-driven monitoring systems and carrier-grade ML infrastructure

By:- P J Krishna Munnaluru

Oracle

Conf42 MLOps

Agenda

01

Current Telecommunications Monitoring Challenges

The limitations of traditional approaches and their impact

02

MLOps Architecture Overview

Core components and integration points with existing systems

03

Neural Network Implementation

Model development, training pipeline, and performance metrics

04

Production Deployment Strategy

Containerization, scaling, and zero-downtime updates

05

Business Impact & Results

Quantifiable improvements across key metrics

The Traditional Monitoring Gap

Current State

Limited Coverage

Only 40% network visibility with blind spots in critical areas

Reactive Approach

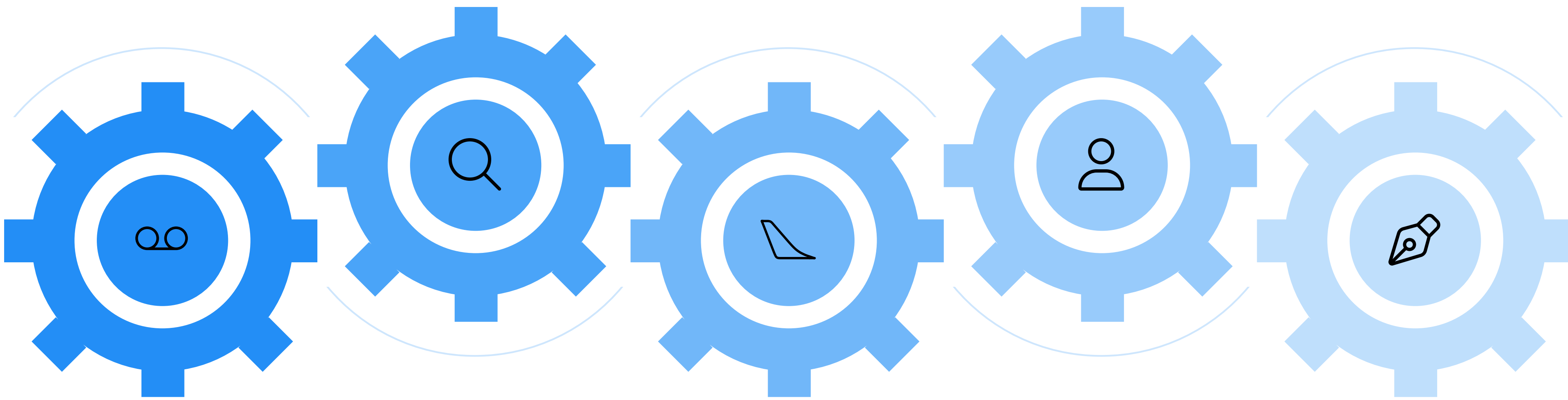
15-30 minute response times to detected issues

Statistical Limitations

65% accuracy in call quality prediction using legacy methods

Traditional monitoring tools struggle with the complexity of modern VoIP infrastructure, resulting in costly delays and customer impact.

MLOps Architecture Overview



Ingest VoIP Streams

Capture call events and audio in real time

Stream via Kafka

Reliable event streaming and buffering

Feature Engineering

Extract features, enrich, and store

Model Training (Kubeflow)

Train, validate, and version models

Registry, Deploy & Monitor

MLflow registry, endpoints, Prometheus & Grafana

Our production-ready architecture integrates data engineering, model training, and operational systems to create a seamless ML lifecycle with built-in governance.

Core MLOps Components

Real-time Data Ingestion

Apache Kafka streams process thousands of network events per second with <100ms latency

Automated Training Pipeline

Kubeflow orchestrates distributed training across GPU clusters with automated hyperparameter tuning

Deployment Automation

Blue-green deployment ensures zero-downtime updates with automated rollback capabilities

Comprehensive Monitoring

Custom Prometheus metrics and Grafana dashboards track model performance and data quality

Each component is containerized and managed through GitOps practices, ensuring consistency across development, staging, and production environments.

Neural Network Implementation

Model Architecture

Our neural network combines CNN layers for feature extraction from packet sequences with LSTM layers for temporal pattern recognition across call sessions.

Key Innovations:

- Custom embedding layer for protocol-specific features
- Attention mechanism that prioritizes anomalous patterns
- Hierarchical feature extraction across multiple time scales
- Ensemble approach combining spectral and time-domain analysis

88% prediction accuracy - 23 percentage points higher than traditional statistical methods

Automated Feature Engineering Pipeline

Raw Packet Collection

High-throughput collectors deployed at network edge capture SIP/RTP metrics and QoS parameters

Real-time Transformation

Apache Beam pipeline processes 10K+ events/second, extracting 45+ features with <200ms latency

Automated Feature Selection

Recursive feature elimination automatically identifies optimal feature sets for each model version

Feature Store Integration

Feast manages feature versioning, serving, and point-in-time consistency for training and inference

All pipeline components are instrumented with comprehensive logging and metrics to ensure data quality and lineage tracking.

Model Training Orchestration

Automated Data Validation

TensorFlow Data Validation ensures schema consistency and detects drift

Distributed Training

Kubeflow pipelines orchestrate GPU-accelerated training across clusters

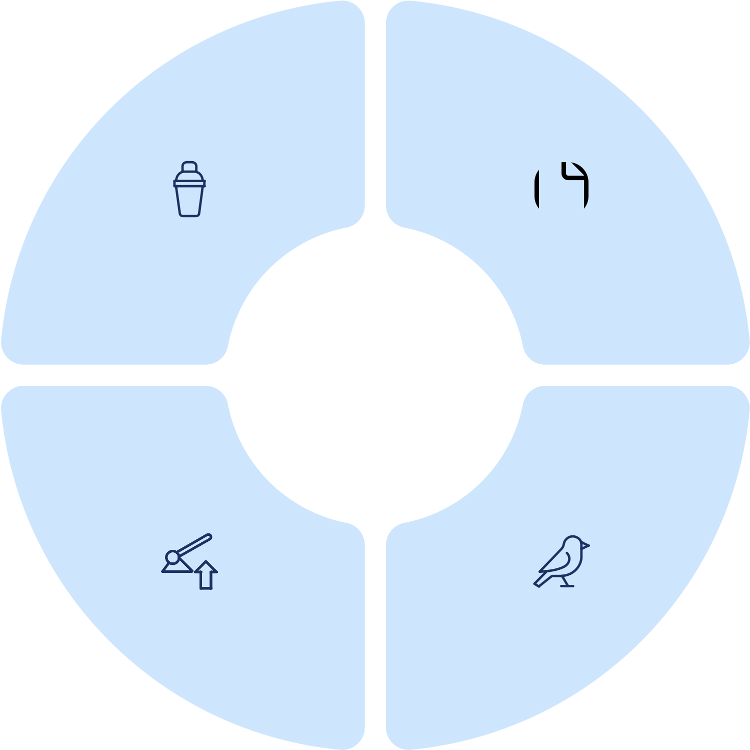
Experiment Tracking

MLflow records all parameters, metrics, and artifacts for reproducibility

Model Registry

Versioned models with approval workflows and lineage tracking

Production Deployment Strategy



Containerization

Models packaged as lightweight containers with optimized TensorFlow Serving

Kubernetes Orchestration

Horizontal pod autoscaling based on traffic patterns and resource utilization

Geographic Distribution

Edge inference nodes deployed across regional data centers for <50ms response time

Blue-Green Deployment

Zero-downtime updates with instant rollback capability if performance degrades

This architecture maintains 96% anomaly detection accuracy while processing thousands of concurrent requests with 99.99% availability SLA.

Comprehensive Monitoring System

What We Monitor

Model Performance

Accuracy, precision, recall tracked against deployed versions

Data Quality

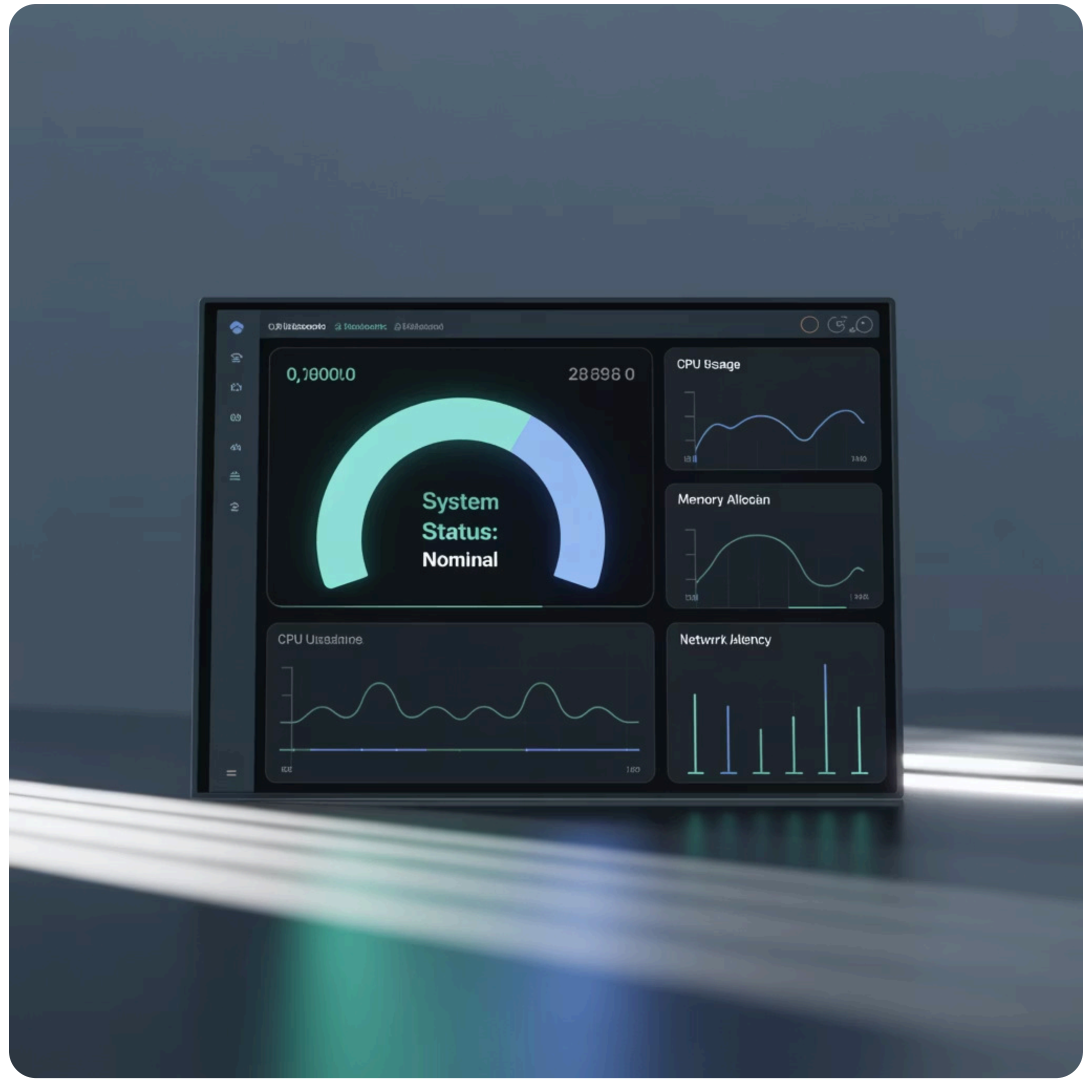
Feature distributions, missing values, schema violations

System Health

Inference latency, throughput, resource utilization

Drift Detection

Statistical tests for concept and data drift with automated alerts



Automated Response Actions

- Model retraining triggered by drift thresholds
- Automated A/B testing for performance evaluation
- Alert escalation with on-call rotation
- Self-healing for common infrastructure issues

Explainability and Governance

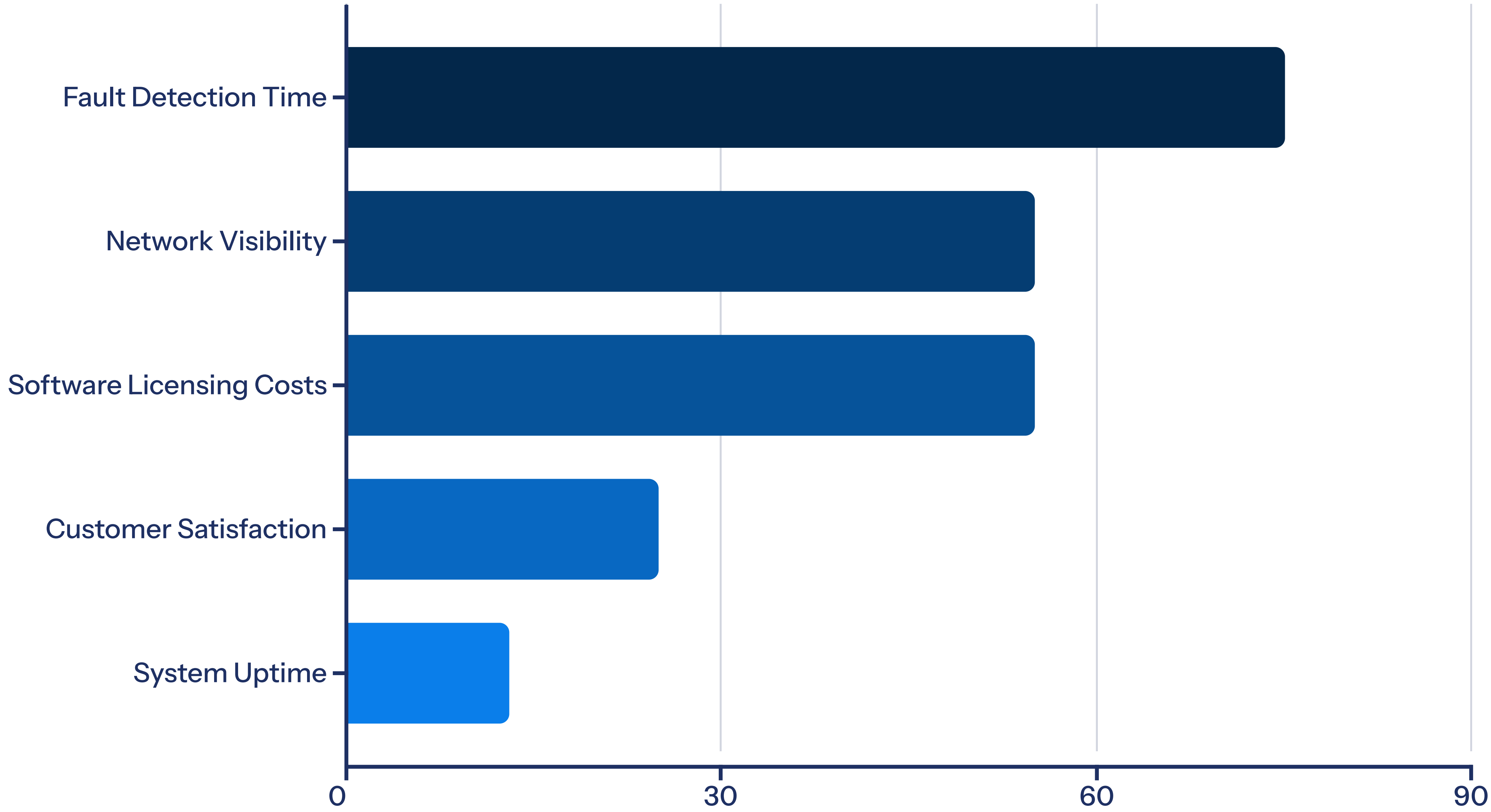
Building Trust Through Transparency

Our MLOps pipeline incorporates robust explainability tools to meet regulatory requirements and build operational trust:

- **SHAP values** identify the most influential features for each prediction
- **Counterfactual explanations** show what changes would alter the outcome
- **Feature attribution dashboards** track importance over time
- **Automated documentation** generates model cards for each version
- **Comprehensive audit logs** track all model training and deployment events

Engineers can trace any prediction back to its causal factors, enabling rapid root cause analysis.

Quantifiable Business Impact



Our MLOps implementation has delivered significant improvements across all key performance indicators, with the most dramatic impact on fault detection time (reduced from 15-30 minutes to just 2-5 minutes).

Implementation Roadmap

Phase 1: Foundation

6-8 weeks

- Data collection infrastructure
- Feature engineering pipeline
- Initial model development
- Basic monitoring setup

Phase 2: Automation

4-6 weeks

- CI/CD pipeline integration
- Automated testing framework
- Model registry implementation
- Enhanced monitoring dashboards

Phase 3: Scale

6-8 weeks

- Distributed training capability
- Geographic deployment
- Advanced explainability tools
- Drift detection mechanisms

Phase 4: Optimization

Ongoing

- Performance tuning
- Model ensemble strategies
- Advanced feature development
- Knowledge transfer & documentation

A phased approach allows for incremental value delivery while building toward the complete MLOps vision.

Key Takeaways

Production-Ready ML Requires End-to-End Thinking

Success depends on integrating data engineering, model development, deployment automation, and operational monitoring into a cohesive system.

Automation Drives Reliability

Automated pipelines for training, testing, deployment, and monitoring are essential for maintaining model performance at scale.

Explainability Builds Trust

Transparent models with clear explanations increase adoption and enable faster troubleshooting.

Thank You