Self-Adaptive Network Optimization with Quantum-Inspired Algorithms

Leveraging Real-Time Data, AI, and Quantum-Inspired Approaches for 5G Network Optimization

<u>Name: B Saran, Bhavesh G, Danush Vikraman G B</u>

Introduction to the Project

- This project focuses on optimizing 5G networks by employing AI-driven predictive analytics and quantum-inspired algorithms. The network management system adapts dynamically in real-time, reducing latency, balancing traffic loads, and improving bandwidth utilization. The key components are:
  - Real-time data collection from network devices and sensors.
  - Predictive analytics with AI models.
  - Optimization using quantum-inspired algorithms.
  - Automated network management.
  - Continuous feedback loops for improvement.

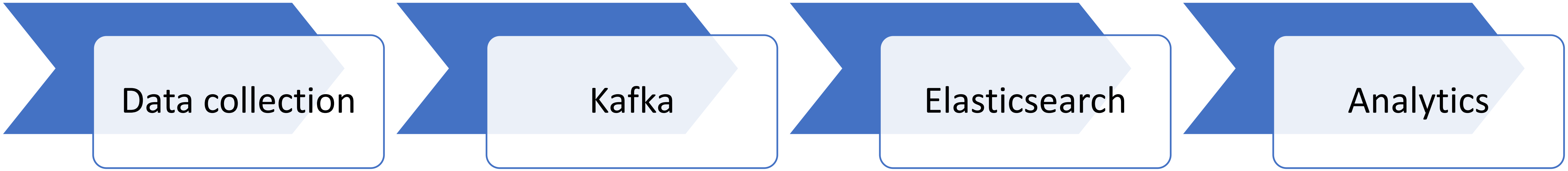
Project Architecture Overview

- The project follows a six-step architecture that integrates real-time data processing, predictive modeling, optimization, and self-adaptive management. The architecture includes:

1. Data collection using Apache Kafka.
2. Predictive modeling with TensorFlow.
3. Quantum-inspired optimization with Ocean SDK.
4. Network automation using ONAP, Ansible, and OpenDaylight.
5. Continuous monitoring and feedback loops with Prometheus, Grafana, and reinforcement learning.
6. Ongoing improvement via model refinement and optimization feedback.

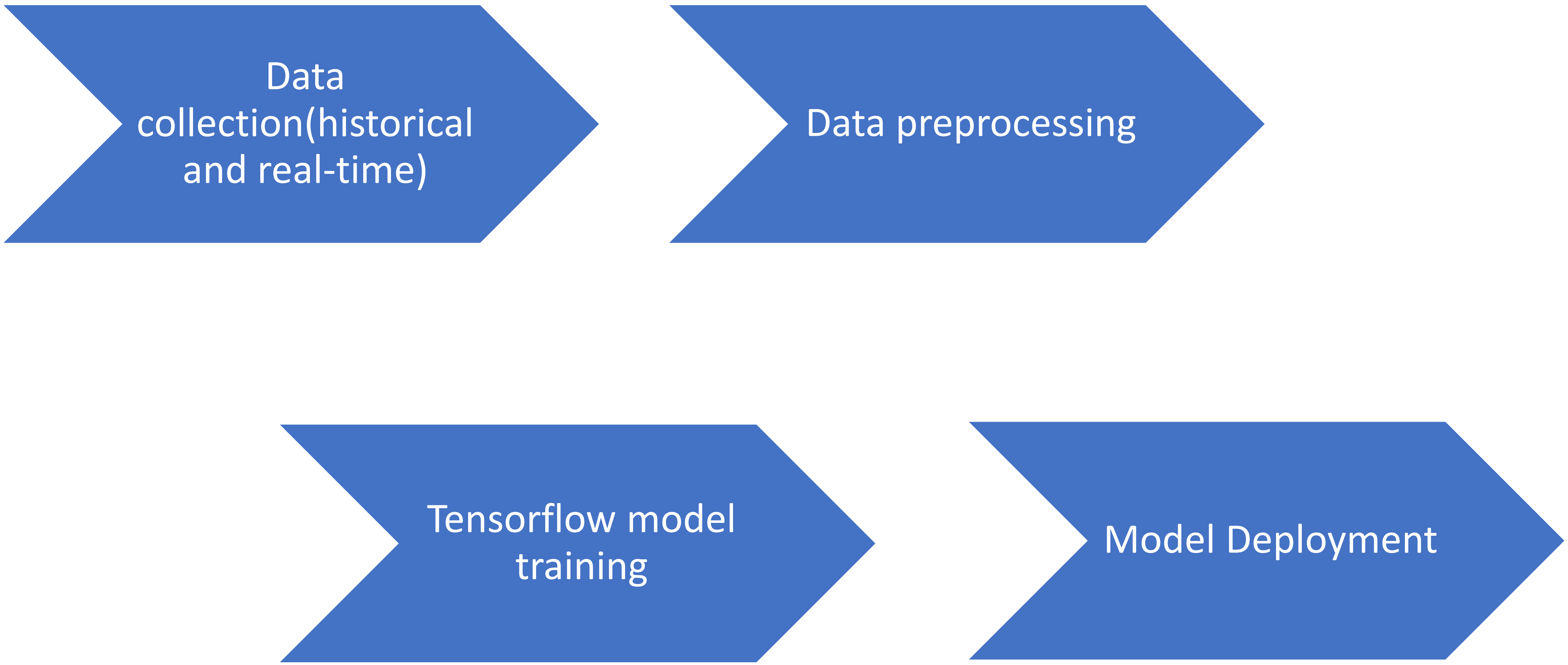
Real-Time Data Collection and Processing

1. **Data Source Setup**: Collect real-time data such as traffic patterns, latency, bandwidth usage, and error rates from network sensors and devices.
2. **Data Ingestion with Kafka**: Use Apache Kafka to handle high-throughput, low-latency data streams from multiple sources. Kafka is deployed on Kubernetes for scalability and resilience, making it ideal for handling 5G network data.
3. **Data Storage**: Store streamed data in Elasticsearch, which allows for real-time indexing and querying, making the data easily accessible for analytics.


Data collection
Kafka
Elasticsearch
Analytics

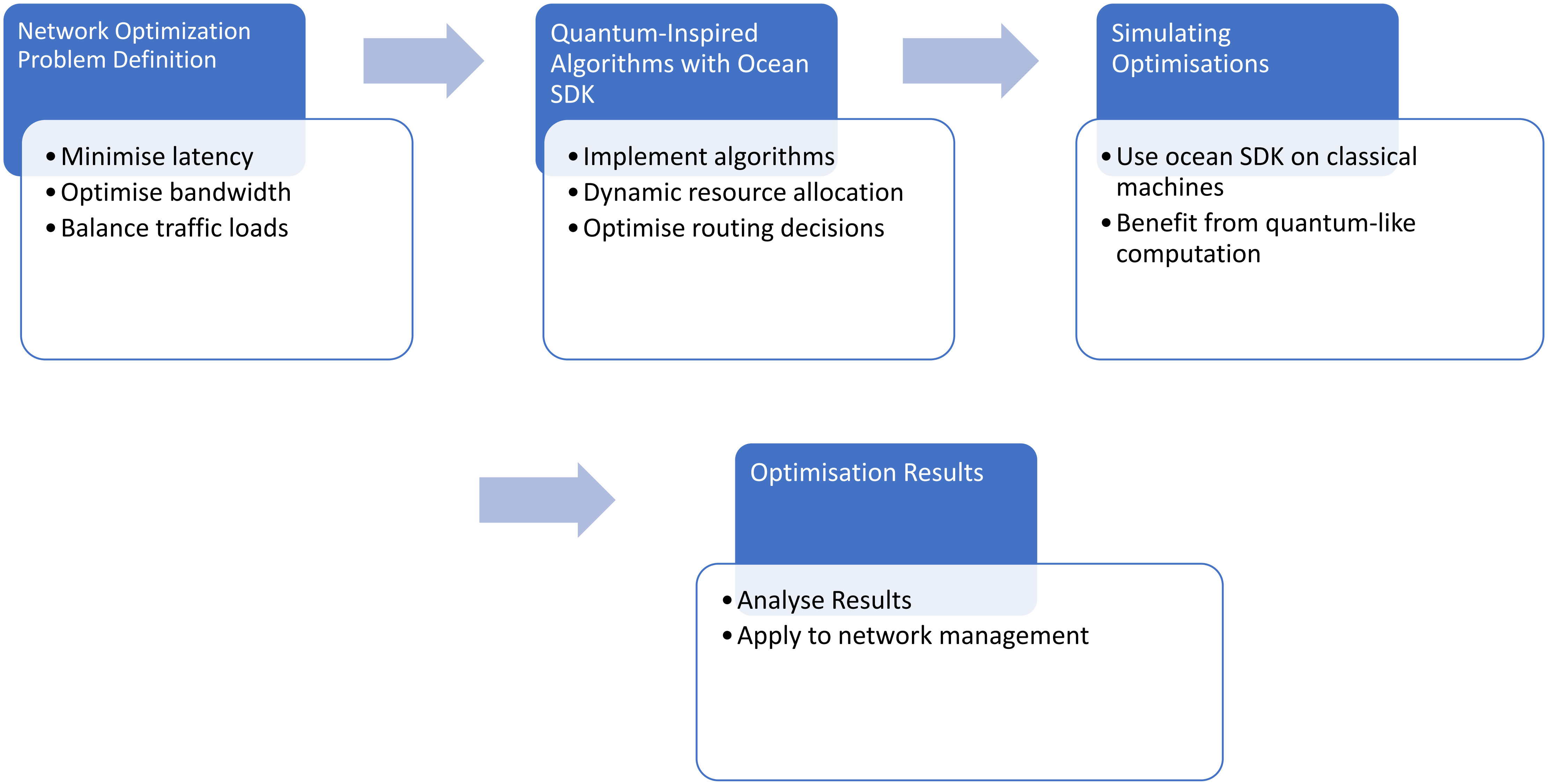
Building Real-Time Predictive Analytics Models

1. **Data Preprocessing**: Process historical and real-time network performance data from Kafka, preparing it for model training.
2. **TensorFlow Model Training**: Train machine learning models in TensorFlow to predict potential network issues, such as bandwidth congestion, latency spikes, and equipment failure. The models continuously learn and improve using real-time data.
3. **Model Deployment on Kubernetes**: Deploy the trained models as Docker containers on Kubernetes, ensuring scalability and high availability for processing real-time data.


Data collection(historical and real-time)
Data preprocessing
Tensorflow model training
Model Deployment

Quantum-Inspired Optimization for Network Efficiency

1. **Network Optimization Problem Definition**: Define network optimization goals, such as minimizing latency, optimizing bandwidth, and balancing traffic loads across the network.
2. **Quantum-Inspired Algorithms with Ocean SDK**: Use Ocean SDK to implement quantum-inspired algorithms. These algorithms help dynamically allocate network resources like bandwidth or optimize routing decisions.
3. **Simulating Optimization**: Perform quantum-inspired optimization on classical machines using Ocean SDK, eliminating the need for quantum hardware while still benefiting from quantum-like computation.


Network Optimization Problem Definition
Minimise latency
Optimise bandwidth
Balance traffic loads
Quantum-Inspired Algorithms with Ocean SDK
Implement algorithms
Dynamic resource allocation
Optimise routing decisions
Simulating Optimisations
Use ocean SDK on classical machines
Benefit from quantum-like computation
Optimisation Results
Analyse Results
Apply to network management

Implementing Self-Adaptive Network Management

1. **ONAP for Orchestration**: Deploy ONAP for automating network service orchestration and lifecycle management. ONAP uses predictive model outcomes and optimization results to make real-time configuration changes.
2. **Ansible for Network Automation**: Use Ansible to automate network management tasks, such as load balancing and failure recovery, based on TensorFlow model predictions.
3. **SDN Control with OpenDaylight**: Integrate OpenDaylight for Software Defined Networking (SDN) control, allowing dynamic adjustment of network routes and configurations based on traffic conditions.

Real-time Monitoring and Feedback Loop for Continuous Learning

1. **Prometheus for Monitoring**: Collect real-time performance metrics like packet loss, latency, and bandwidth usage using Prometheus. Alerts are configured to trigger corrective actions when performance issues arise.
2. **Grafana for Visualization**: Create real-time dashboards using Grafana, which visualizes performance metrics and predictive model outcomes, offering an easy-to-understand view of network health.
3. **Reinforcement Learning for Feedback**: Implement a feedback loop using OpenAI Gym and Reinforcement Learning (RL). The RL agent adjusts network configurations over time based on real-time data and learned outcomes to continuously optimize performance.

Continuous Model Refinement and System Optimization

1. **Model Updates with TensorFlow**: Continuously retrain the TensorFlow models with new real-time data to improve predictive accuracy and adjust to evolving network conditions.
2. **Feedback-Driven Optimization**: Use data collected by Prometheus to refine system actions. This feedback informs ONAP and Ansible automation rules for ongoing optimization.
3. **Visual Monitoring with Grafana**: Track improvements in network performance and system evolution through real-time dashboards.

Key Benefits:

- Dynamic network optimization using AI and quantum-inspired algorithms.
- Real-time predictions for issue resolution.
- Automated network reconfiguration.
- Continuous improvement based on real-time data.

Thank You