



# Rakshak

ENTERPRISE LOG ANOMALY DETECTION SYSTEM

[log-anomaly-frontend.vercel.app](https://log-anomaly-frontend.vercel.app)

Krishna Sharma  
AP22110010128

# Introduction

Enterprise Log Anomaly Detection Platform: Production-grade AI system that identifies anomalies across diverse enterprise logs with real-time, high-precision detection.

## Key Outcomes

- 88.5% F1-Score | 0.94 AUROC | 91.2% Balanced Accuracy
- <200ms inference latency across 16 log sources
- Handles extreme imbalance up to 249:1
- 32,000+ logs processed with cross-source generalization

## Core Intelligence

- 848-dimensional hybrid features (BERT + templates + stats)
- 22 trained models (ML, DL, BERT, Meta, Federated)
- LOSO evaluation → robust cross-domain performance

## Deployment Ready

- REST API (Django) + React UI + HuggingFace models
- Cloud-hosted (Render + Vercel) with real-time anomaly preview

# Problem & Motivation

The Enterprise Log Challenge: Modern distributed systems generate millions of logs per day, across multiple sources, formats, and platforms. Hidden within them are rare but critical anomalies: breaches, failures, outages, misuse.

## Core Pain Points

- Extreme class imbalance (as high as 249:1)
- Multi-source heterogeneity
- High-volume, real-time ingestion
- Unstructured & semi-structured patterns
- Evolving system behavior

## Why Traditional Methods Fail

- Rule-based SIEMs require constant tuning and still miss novel events
- Standard ML assumes balanced data → overfits to “normal”
- Many systems cannot transfer learn across log sources
- Deep models without imbalance awareness spike false negatives

## Motivation

- To build a resilient, interpretable, real-time anomaly detection engine that:
- Handles severe imbalance without collapsing recall
  - Generalizes across 16 distinct log environments
  - Detects never-seen-before behaviors
  - Delivers predictions within <200ms

# Solution Overview

What This System Delivers: A production-grade anomaly detection platform that automates log intelligence, handles extreme imbalance, and scales to multi-source enterprise environments.

## Core Capabilities

- Detects anomalies across 16 independent log ecosystems
- Learns semantic + structural patterns via BERT + templates + statistics
- <200ms real-time inference for streaming operational reliability

## End-to-End Architecture

Raw Logs → Labeling → Processing → Feature Engineering → Models → Inference → UI

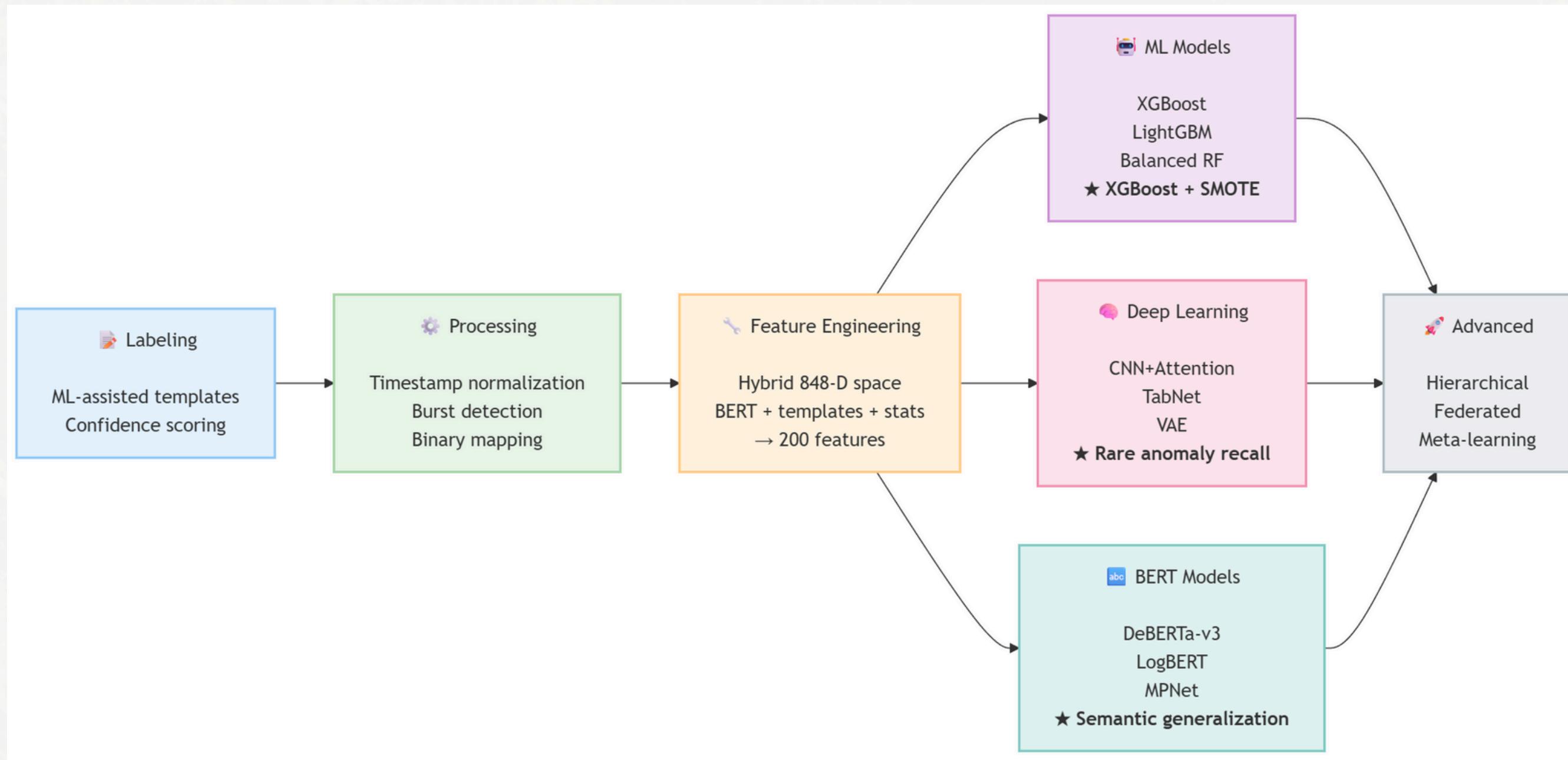
- Labeling: Smart ML-assisted template labeling
- Processing: Temporal & burst normalization for 16 sources
- Feature Engineering: 848-D hybrid representations (BERT, Drain, stats)
- Modeling: ML, DL, BERT, Federated, Hierarchical
- Deployment: Django REST API + React UI + Cloud serving

## What Makes It Effective

- Cross-source generalization (LOSO): learns beyond single dataset boundaries
- Multi-level imbalance strategy: SMOTE + Focal Loss + threshold tuning
- Hybrid feature stack: semantic + structural + temporal insights
- Model ensemble strategy: choose best model per environment

# Complete Pipeline Flow

Transforms raw logs into real-time anomaly decisions



# Data Landscape

True anomalies are rare, hidden, and format-dependent—demanding a cross-source, imbalance-aware approach.

## Multi-Source Enterprise Logs

- 32,000+ logs collected across 16 heterogeneous sources
- Formats include: Apache, Linux, Android, HDFS, OpenSSH, Hadoop, Spark, HealthApp, Cloud services

## Diversity & Variability

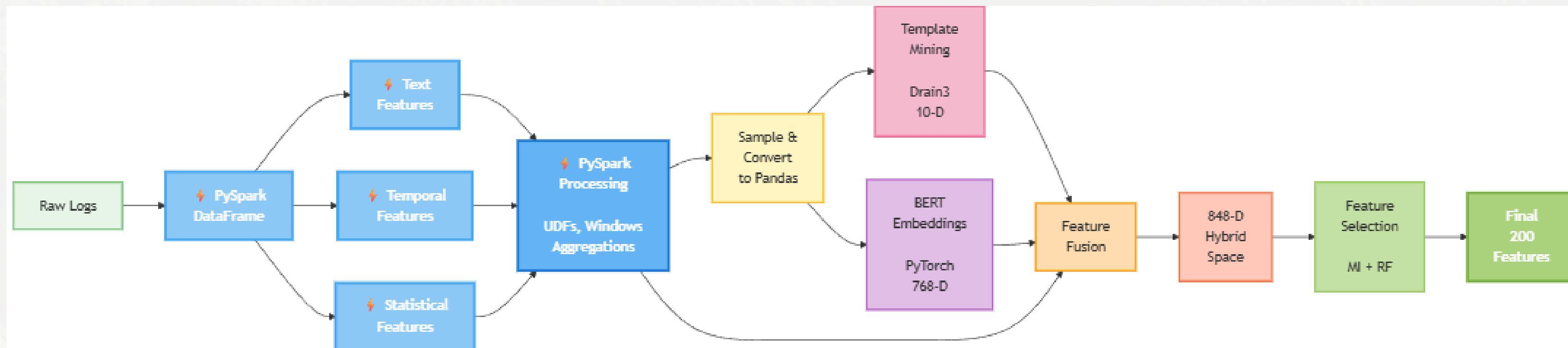
- Structured + semi-structured + free-text logs
- Varying grammar, templates, verbosity, timestamp styles
- Frequent out-of-vocabulary tokens (IP, UUID, HEX, PATH)

## Severe Class Imbalance

- Anomalies often <1% of total
- Worst-case ratio: 249 : 1 (Spark, HealthApp)
- Standard classifiers collapse to “all normal” without correction

# Feature Engineering

Distributed Feature Pipeline (PySpark + BERT): A scalable feature workflow transforming raw logs into a 200-feature anomaly-ready vector.



## PySpark's Role

Parallel log processing + temporal bursts + statistical windows using UDFs & Window functions.

- Drain3 Templates (10-D) → structure patterns
- BERT Embeddings (768-D) → semantic meaning
- PySpark Features (~70+ signals) → bursts, entropy, frequency, off-hours
- MI + RF Selection → final 200 best features

# Model Architecture Overview

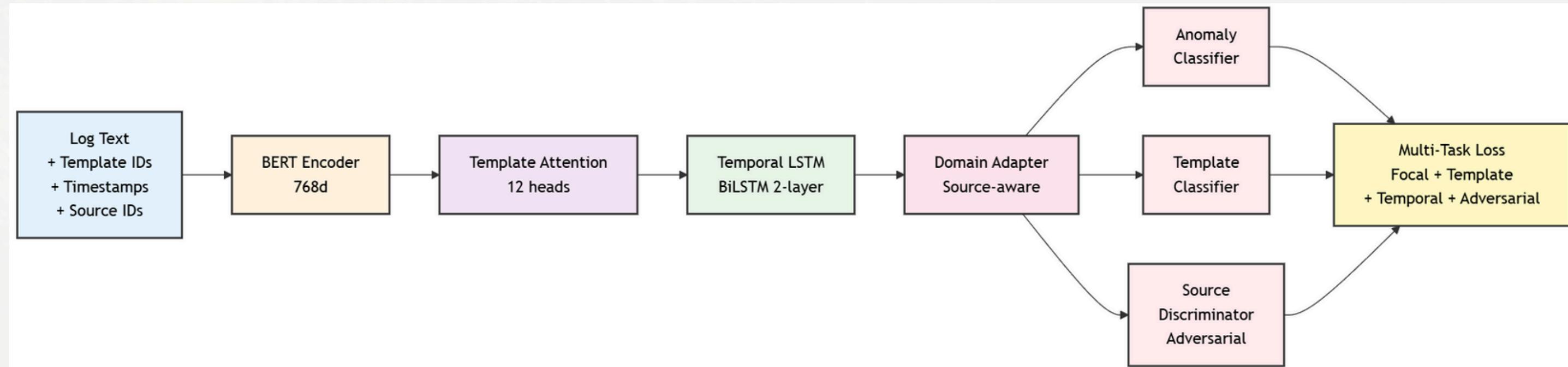
Multi-stack system evaluated across 16 log sources for cross-domain robustness.

Category	Models	Purpose
ML	XGBoost, LightGBM, RF, SVM, Balanced RF, Easy Ensemble	Fast, production-grade baseline
DL	CNN+Attention, VAE, TabNet, FLNN	Handles imbalance, hidden patterns
BERT	LogBERT, DeBERTa-v3, MPNet, DAPT-BERT	Deep semantic log understanding
Advanced	Hierarchical Transformer, Federated Contrastive, Meta-Learner	Cross-source & few-shot generalization

## Why Multi-Model?

- Diverse logs require both structural + semantic learners
- BERT captures intent, ML handles speed, DL absorbs imbalance
- Advanced models extend to new unseen log sources

# Hierarchical Log Transformer



**Problem: Logs Have Multi-Level Structure**

- Token-level: Individual words ("ERROR", "timeout")
- Template-level: Similar patterns ("Connection failed <IP>")
- Sequence-level: Temporal relationships
- Source-level: Cross-domain variations

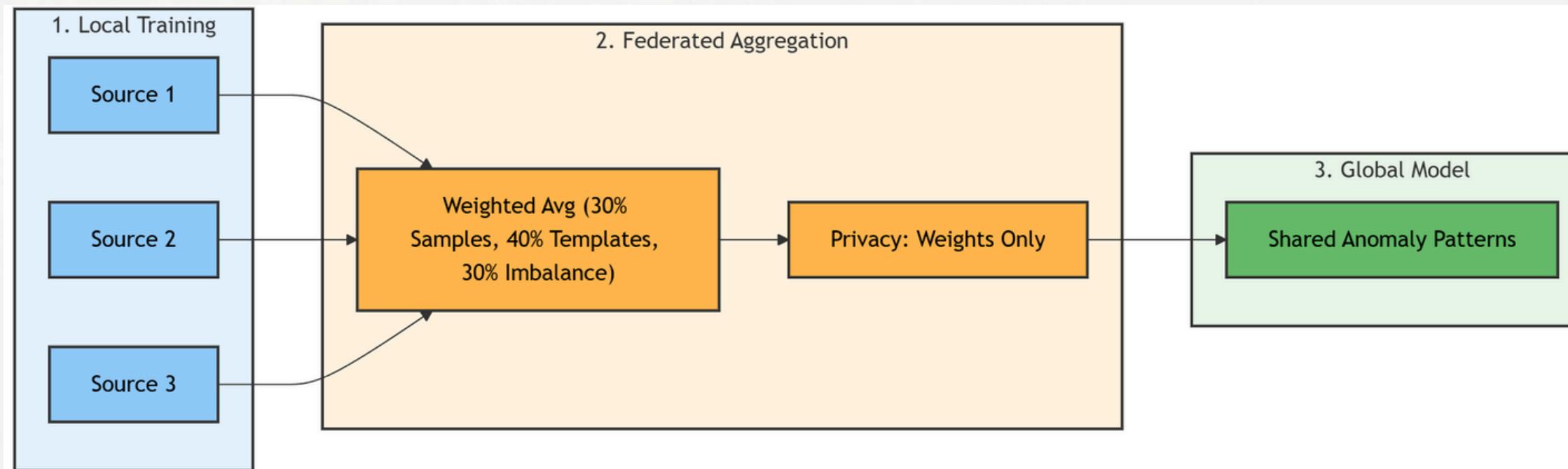
## Novel Components:

- Template-Aware Attention: Attention bias based on Drain3 templates which groups semantically similar logs before classification
- Temporal Consistency Loss: Penalizes contradictory predictions in time windows
- Cross-Source Adversarial Training: Domain-invariant features via gradient reversal

## Results

- F1-Macro: 86% ( $\pm 11\%$ ) across 13 sources
- Key Win: Handles Apache (3:1) to Android (73:1) imbalance
- Speed: <200ms inference with frozen BERT layers

# Federated Contrastive Learning



## Problem: Privacy + Cross-Source Learning

- Each log source = sensitive client (can't share raw data)
- Need to learn from 16 sources without centralizing data
- Extreme class imbalance varies by source (3:1 to 332:1)

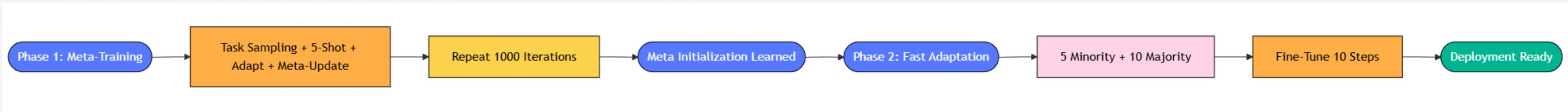
## Novel Components:

- Template-Based Contrastive Pairs: Use Drain3 to create semantic augmentations
- Imbalance-Aware FedAvg: Weight clients by  $1/\text{imbalance\_ratio}$
- Template Alignment: Align class prototypes across sources

## Results

- F1-Macro: 40% ( $\pm 15\%$ ) across 13 sources
- Privacy: Zero raw data sharing (federated)
- Convergence: 10 rounds (vs 100+ for standard FL)

# Meta-Learning



## Problem: New Log Sources with <10 Labeled Samples

- Real-world scenario: New application deployed, only 5-8 anomalies labeled
- Challenge: Extreme imbalance (Spark: 8 anomalies in 2000 logs = 249:1)
- Goal: Adapt to new source in <5 minutes

## Novel Components:

- Curriculum Learning: Train on balanced sources first (3:1) → extreme (249:1)
- Adaptive Inner Steps: 10 steps (moderate) → 30 steps (extreme imbalance)
- Focal Loss Meta-Objective:  $\alpha=0.75$ ,  $\gamma=3.0$  (tuned per imbalance ratio)

## Results

- F1-Macro: 94.2% ( $\pm 6\%$ ) - BEST OVERALL
- Balanced Acc: 96.97% | AUROC: 99.20% | MCC: 88.48%
- Fast Adaptation: 5-shot → 90%+ F1 in <2 minutes
- Generalization: Works across unseen sources (LOSO validation)

**Research Contribution:** First meta-learning approach for log anomaly detection with curriculum learning and adaptive inner-loop strategies for extreme imbalance

# Deployment Architecture

End-to-end real-time anomaly detection delivered through a modular web stack.

## Frontend

React + Tailwind

- Live anomaly feed
- Confidence scoring
- Source auto-identification

## Backend

Django REST API + HuggingFace Model

Serving

- Fast inference routing
- JSON prediction endpoints
- Stateless microservice design

## Cloud Hosting

- UI: Vercel
- API: Render
- Models: HuggingFace Inference Hub



# The End

THANK YOU FOR LISTENING

Krishna Sharma  
AP22110010128