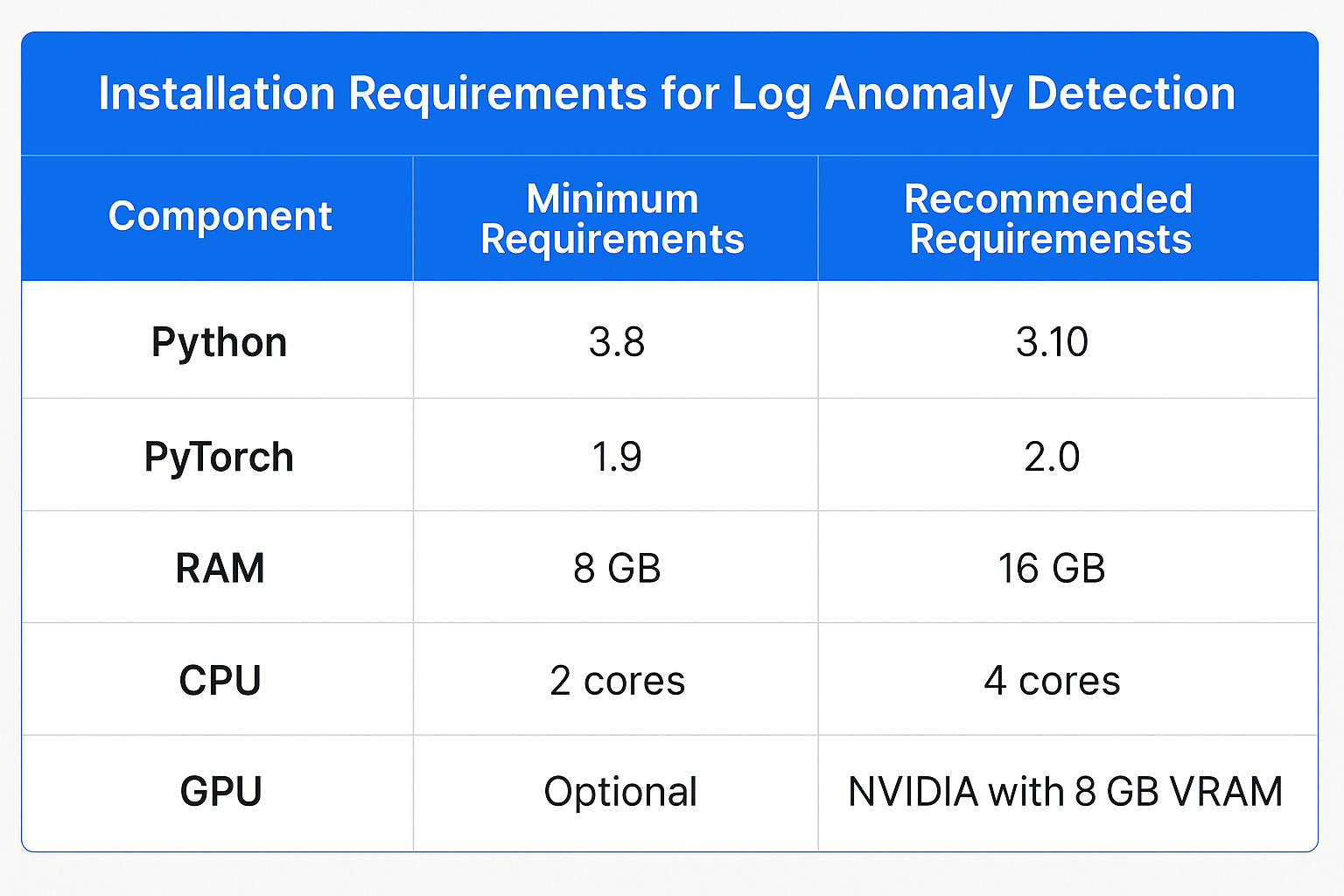
**Log Anomaly Detection System Documentation**

**System Overview**: An advanced ensemble-based log anomaly detection system utilizing Hybrid Attention LSTM Autoencoders for comprehensive security monitoring and operational intelligence in Linux environments.

**1. Quick Start Guide**

**1.1 Installation and Setup**



**Installation Steps:**

# Clone repository  
git clone https://github.com/AK11105/AI-driven-SIEM-System.git  
cd log-anomaly-detection  
  
# Install dependencies  
pip install -r requirements.txt  
  
# Initialize configuration  
python setup.py --init-config

**1.2 Basic Usage**

**Training a Model:**

from model import HybridEnsembleDetector  
  
# Load and preprocess data  
detector = HybridEnsembleDetector()  
detector.train('data/logs/processed/Linux.csv')  
  
# Generate deployment package  
detector.save\_deployment\_package('model.pkl')

**Running Detection:**

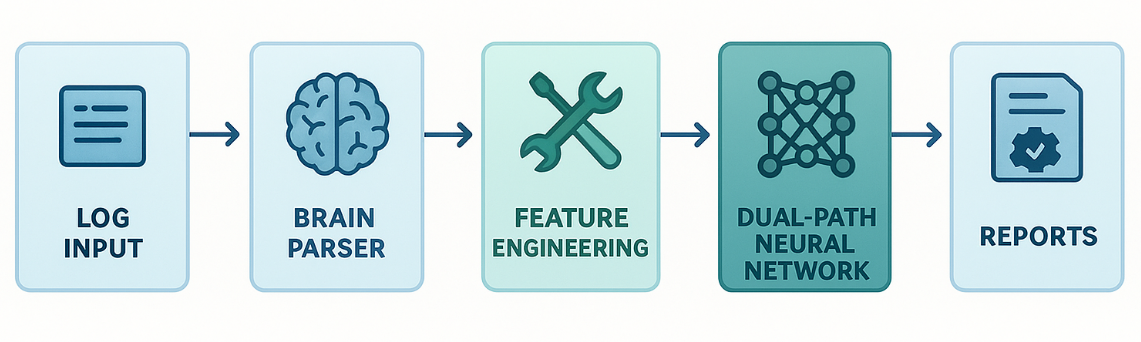
# Load trained model  
detector = HybridEnsembleDetector.load('model.pkl')  
  
# Detect anomalies  
results = detector.predict('new\_logs.csv')  
detector.generate\_report(results, 'anomaly\_report.json')

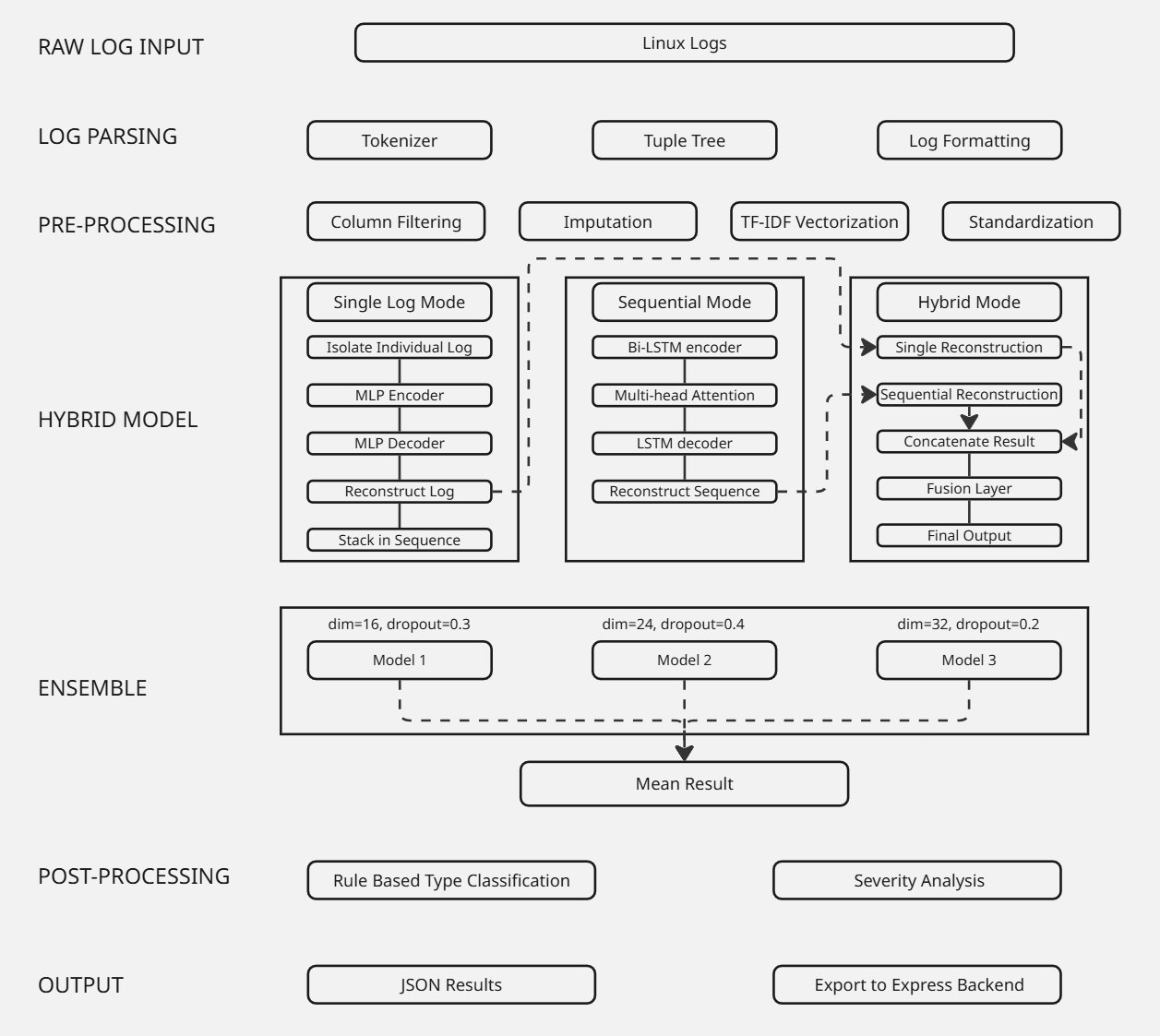
**2. System Architecture**

**2.1 Overview and Core Functionality**

The system combines unsupervised machine learning with rule-based classification to achieve comprehensive log anomaly detection. It processes logs through two complementary pathways: individual log analysis for content-based anomalies and sequential analysis for temporal pattern detection.

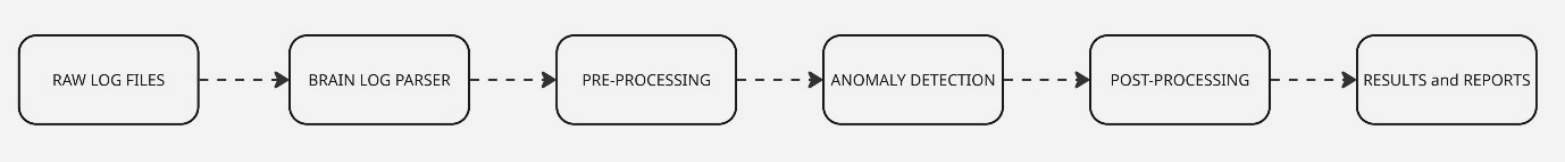
Anomaly detection systems operate on the principle that normal system behavior follows predictable patterns, while anomalous events deviate significantly from these established baselines. The dual-pathway architecture addresses two fundamental types of anomalies: point anomalies (individual outlier log entries) and collective anomalies (sequences that appear normal individually but are anomalous when considered together).



**Core Components:**

* **Brain Log Parser: Converts raw logs to structured format using pattern recognition algorithms that identify static and dynamic content within log messages.**
* **Hybrid Neural Network: Dual-path LSTM-Autoencoder ensemble that processes both sequential dependencies and individual log semantics.**
* **Rule-Based Classifier: Semantic anomaly categorization using expert-defined patterns for domain-specific knowledge integration.**
* **Severity Manager: Dynamic threshold learning and scoring based on statistical distribution analysis.**

**2.2 Data Flow Pipeline**

 The pipeline processes logs through five main stages, each incorporating specific theoretical principles from machine learning and natural language processing:

1. **Parsing:** Raw logs → Structured CSV (EventTemplate, Content, Component) using tokenization and pattern extraction techniques.
2. **Feature Engineering:** Text → Numerical features (TF-IDF, categorical encoding) employing information retrieval and statistical methods.
3. **Neural Processing:** Features → Reconstruction errors (dual-path analysis) using autoencoder theory for unsupervised anomaly detection.
4. **Classification:** Errors → Anomaly types (rule-based semantic analysis) combining statistical thresholds with domain expertise.
5. **Reporting:** Classifications → Structured reports (JSON output) with confidence scores and severity assessments

**3. Data Processing and Feature Engineering**

**3.1 Log Preprocessing Pipeline**

The load\_and\_preprocess function transforms heterogeneous log data into uniform numerical representations while preserving semantic meaning. This transformation is critical because machine learning algorithms require numerical input, while log data is inherently textual and unstructured.

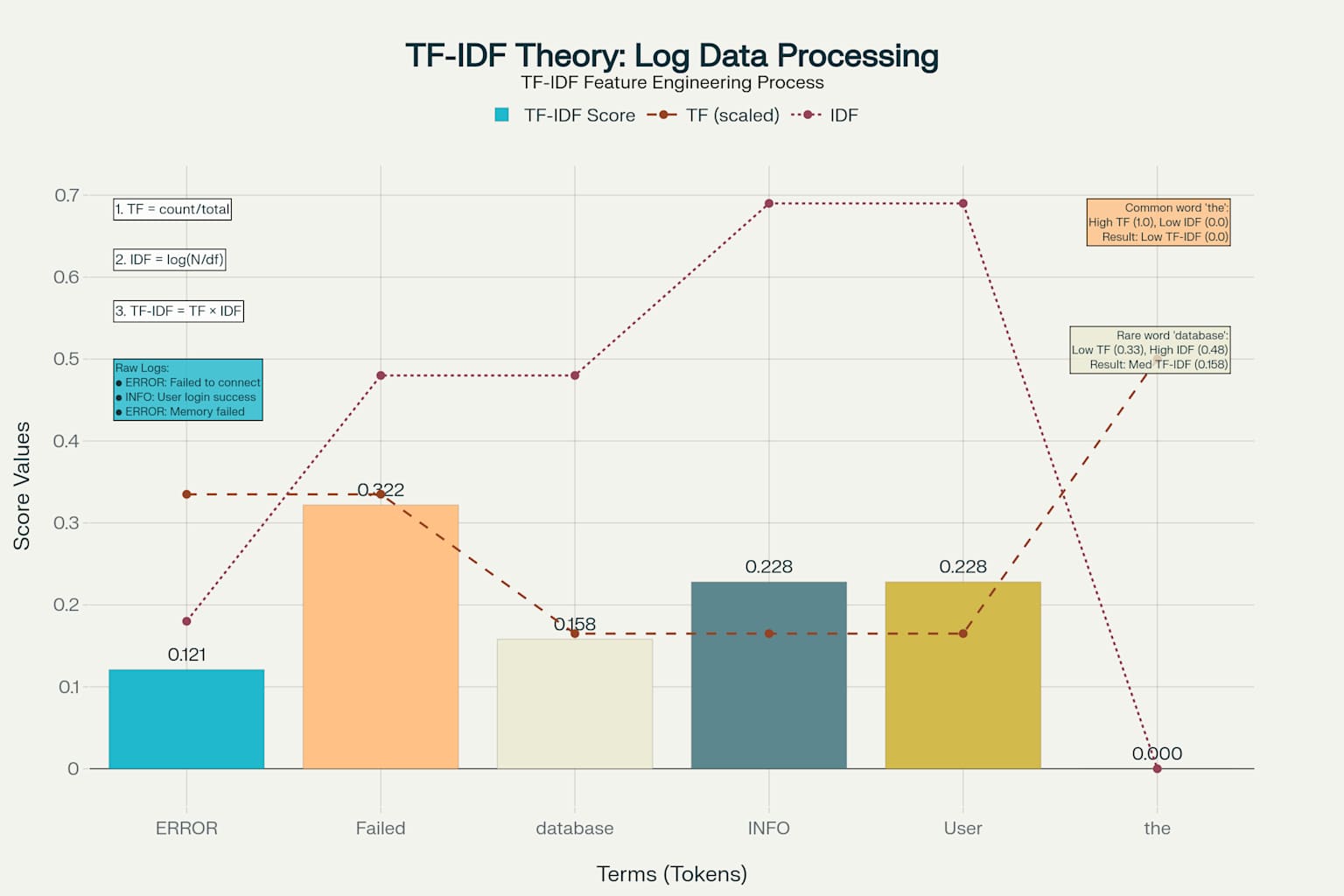
**Feature Categories:**

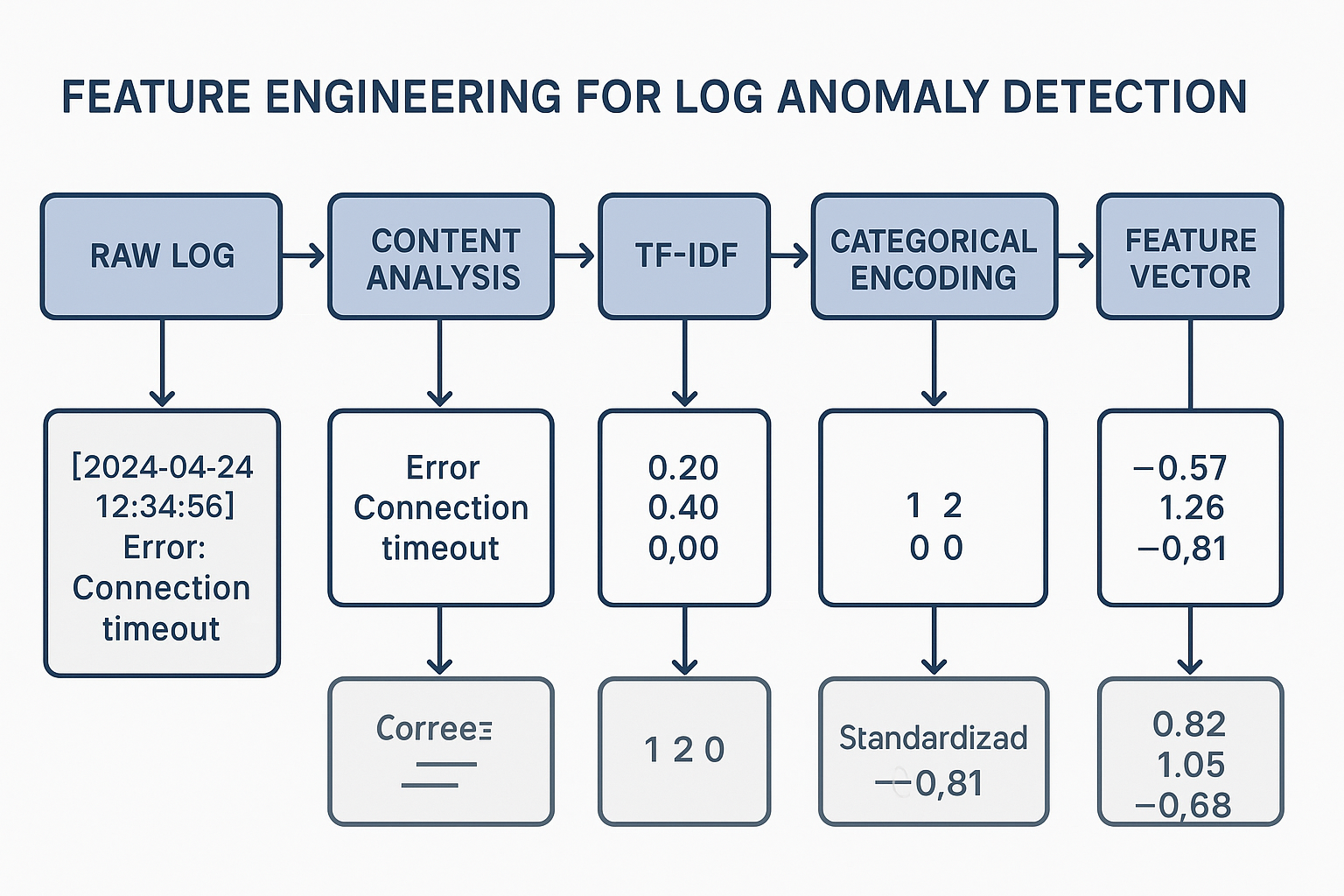
* **Content Features**: Length, word count, semantic indicators (error, warning, critical) based on statistical text analysis.
* **Template Features**: TF-IDF vectorization with log-specific stop words using information retrieval principles where term frequency measures local importance and inverse document frequency measures global rarity.
* **Categorical Features**: Component encoding with cardinality handling using one-hot encoding for nominal variables.
* **Numerical Features**: Standardized using z-score normalization to ensure all features have comparable scales and prevent bias toward high-magnitude features

**Key Preprocessing Steps:**

# Remove temporal identifiers to prevent time-based bias  
excluded\_columns = ['Time', 'Date', 'Month', 'PID', 'LineId']  
  
# Extract semantic features from content  
content\_features = {  
 'content\_length': len(content),  
 'content\_word\_count': len(content.split()),  
 'content\_has\_error': bool(re.search(r'\berror\b', content, re.I)),  
 'content\_has\_warning': bool(re.search(r'\bwarning\b', content, re.I))  
}  
  
# TF-IDF vectorization for event templates  
tfidf = TfidfVectorizer(max\_features=50, min\_df=3, max\_df=0.7)  
template\_features = tfidf.fit\_transform(event\_templates)

The TF-IDF approach is particularly effective for log data because it emphasizes rare but informative terms while de-emphasizing common but less discriminative terms. The TF-IDF implementation incorporates custom stop word lists specifically tailored for log data, removing common but uninformative terms such as prepositions, articles, and generic system terminology.





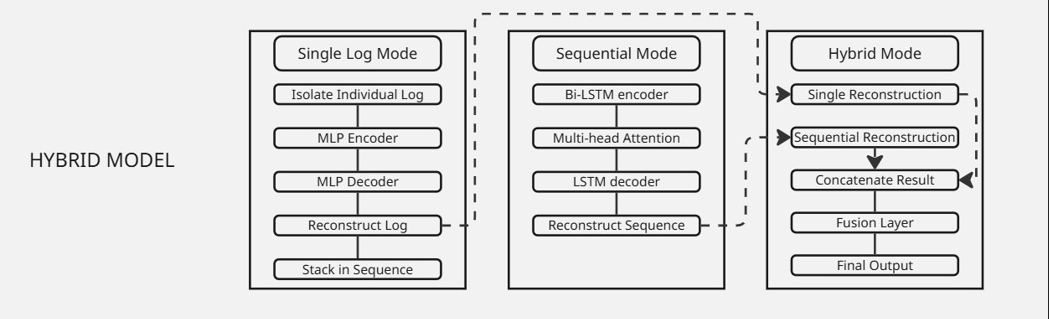
**3.2 Sequence Generation**

The LogDataset class implements sliding window approach for temporal analysis:

* **Window Size: 8 logs (configurable) - selected to capture short-term temporal dependencies while maintaining computational efficiency**
* **Stride: 8 logs (non-overlapping, configurable) - prevents data leakage between training and validation sequences**
* **Format: PyTorch tensors for GPU acceleration enabling parallel processing of multiple sequences**

**4. Neural Network Architecture**

**4.1 Hybrid Attention LSTM Autoencoder**

****The core model implements a dual-path architecture addressing both sequential and individual log analysis, combining the strengths of recurrent neural networks for temporal modeling with multi-layer perceptrons for individual instance analysis.

LSTM autoencoders are designed to learn compressed representations of sequential data through an encoder-decoder architecture. The encoder maps input sequences to a fixed-length latent representation, while the decoder reconstructs the original sequence from this representation. Anomalies are detected by measuring reconstruction error - the assumption being that the model trained on normal data will poorly reconstruct anomalous sequences.

**Sequential Path:**

* **Bidirectional LSTM encoder (hidden\_dim: 16/24/32):** Processes sequences in both forward and backward directions, allowing the model to capture both past and future context for each time step. This bidirectional processing is crucial for understanding log sequences where future context can disambiguate current events.
* **Multi-head self-attention (4 heads):** Implements the attention mechanism to focus on relevant parts of the input sequence, computing attention weights that reflect the relative importance of each sequence element. Multiple attention heads allow the model to attend to different types of relationships simultaneously.
* **LSTM decoder with batch normalization:** Reconstructs the original sequence while batch normalization stabilizes training by normalizing layer inputs to have zero mean and unit variance

**Single Log Path:**

* **Multi-layer perceptron encoder:** Processes individual logs without temporal context, using progressive dimensionality reduction to extract salient features.
* **Progressive dimensionality reduction:** Gradually reduces feature space to force the model to learn efficient representations.
* **Reconstruction through decoder layers:** Reverses the encoding process to reconstruct original input features.

**Fusion Layer:**

* **Linear combination of both paths:** Combines outputs from sequential and single-log pathways using learnable weights.
* **Learned weighting parameters:** Automatically determines optimal combination of both processing modes during training.
* **Consistency loss regularization:** Ensures both pathways produce compatible representations by penalizing large differences between their outputs.

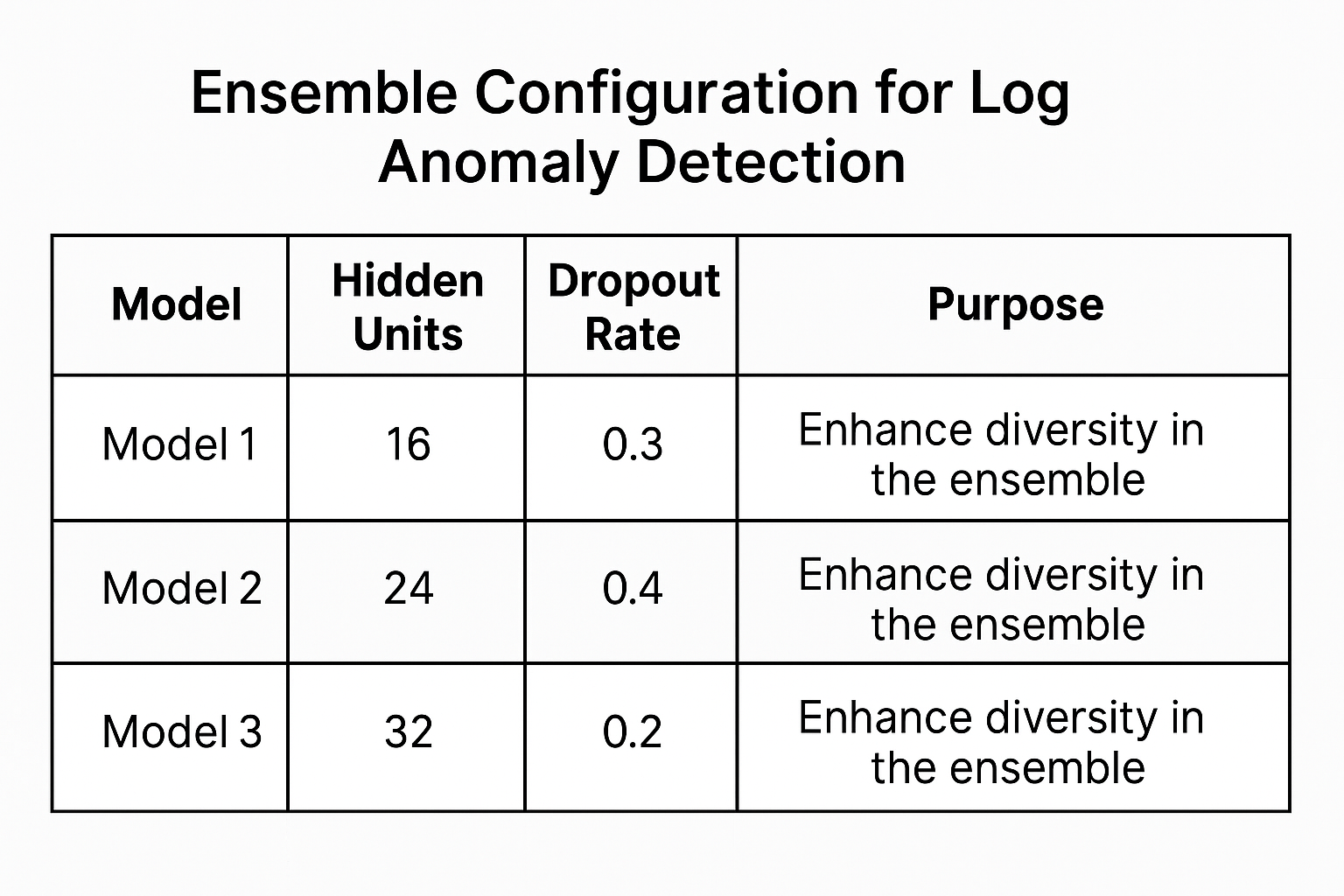
**4.2 Operational Modes**

* **Sequential Mode: Temporal anomaly detection and attack pattern recognition by analyzing sequences of logs to identify patterns that emerge over time.**
* **Single Mode: Real-time individual log analysis for content-based anomalies, enabling immediate detection of problematic log entries.**
* **Hybrid Mode: Comprehensive detection combining both approaches for maximum coverage and accuracy.**

**5. Ensemble Training Strategy**

**5.1 Ensemble Configuration**

Three diverse model configurations ensure robust detection through the principle of ensemble learning, which states that combining multiple models can achieve better performance than any individual model. Ensemble learning addresses the bias-variance tradeoff in machine learning by combining predictions from multiple models trained with different configurations. This approach reduces overfitting (variance) while maintaining predictive power by leveraging model diversity. The effectiveness stems from the principle that individual models make different types of errors, and combining them reduces overall error rate.



**5.2 Training Process**

**Loss Function:**

The loss function combines reconstruction accuracy with consistency between processing pathways. The reconstruction loss measures how well the autoencoder can recreate input sequences, while consistency loss ensures both pathways learn compatible representations.

total\_loss = reconstruction\_loss + 0.1 \* consistency\_loss  
consistency\_loss = MSE(single\_reconstruction, sequential\_reconstruction)

**Optimization:**

* **Adam optimizer (lr=1e-3, weight\_decay=1e-4):** Adaptive moment estimation that combines momentum with adaptive learning rates for each parameter, particularly effective for sparse gradients common in text processing.
* **ReduceLROnPlateau scheduling:** Dynamically reduces learning rate when validation loss plateaus, allowing fine-tuning of parameters when learning stagnates.
* **Early stopping (patience=5 epochs):** Regularization technique that prevents overfitting by stopping training when validation performance stops improving.
* **Gradient clipping (max\_norm=1.0):** Prevents exploding gradient problem common in recurrent networks by limiting gradient magnitude during backpropagation

**5.3 Performance-Based Weighting**

Ensemble weighting utilizes inverse validation loss to assign higher weights to better-performing models during inference. This dynamic weighting approach ensures that ensemble predictions are dominated by the most accurate models while still benefiting from the regularization effects of model diversity.

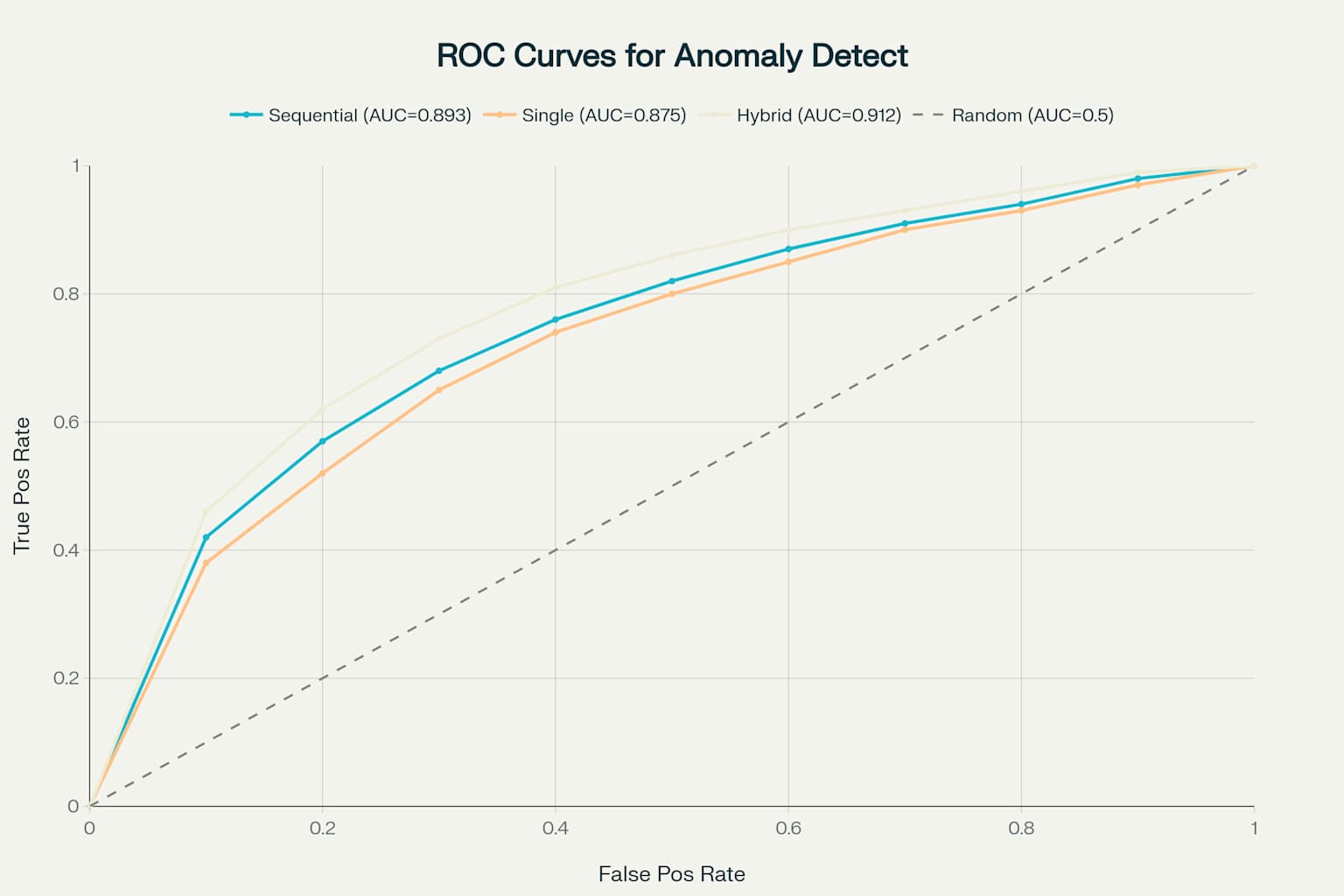
The performance-based weighting system automatically adjusts to account for varying model effectiveness across different types of anomalies. Models that excel at detecting specific anomaly categories receive higher influence for relevant predictions while maintaining ensemble diversity.

Dynamic weight updates enable the ensemble to adapt to changing data characteristics over time. As individual models demonstrate varying performance on new data patterns, the weighting system automatically adjusts to maintain optimal ensemble performance.

weight\_i = (1/loss\_i) / sum(1/loss\_j for all j)  
ensemble\_error = sum(weight\_i \* error\_i for all i)

The weighted combination process maintains computational efficiency while providing robust anomaly detection across diverse log patterns and operational conditions. The averaging mechanism reduces the impact of individual model errors while amplifying consistent anomaly signals across ensemble members.

Error aggregation includes validation of ensemble predictions to ensure that weighted averaging produces reasonable anomaly scores. The system monitors ensemble performance and adjusts weighting strategies if individual models begin producing inconsistent results.



**6. Anomaly Classification and Severity Assessment**

**6.1 Rule-Based Classification**

Rule-based classification systems use expert-defined patterns to categorize data based on domain knowledge. This approach complements machine learning by incorporating human expertise and providing interpretable classifications

Six anomaly categories identified through regex patterns in the log message content namely:

* Memory errors
* Authentication failures
* Filesystem issues
* Network problems
* Permission violations
* Critical system events

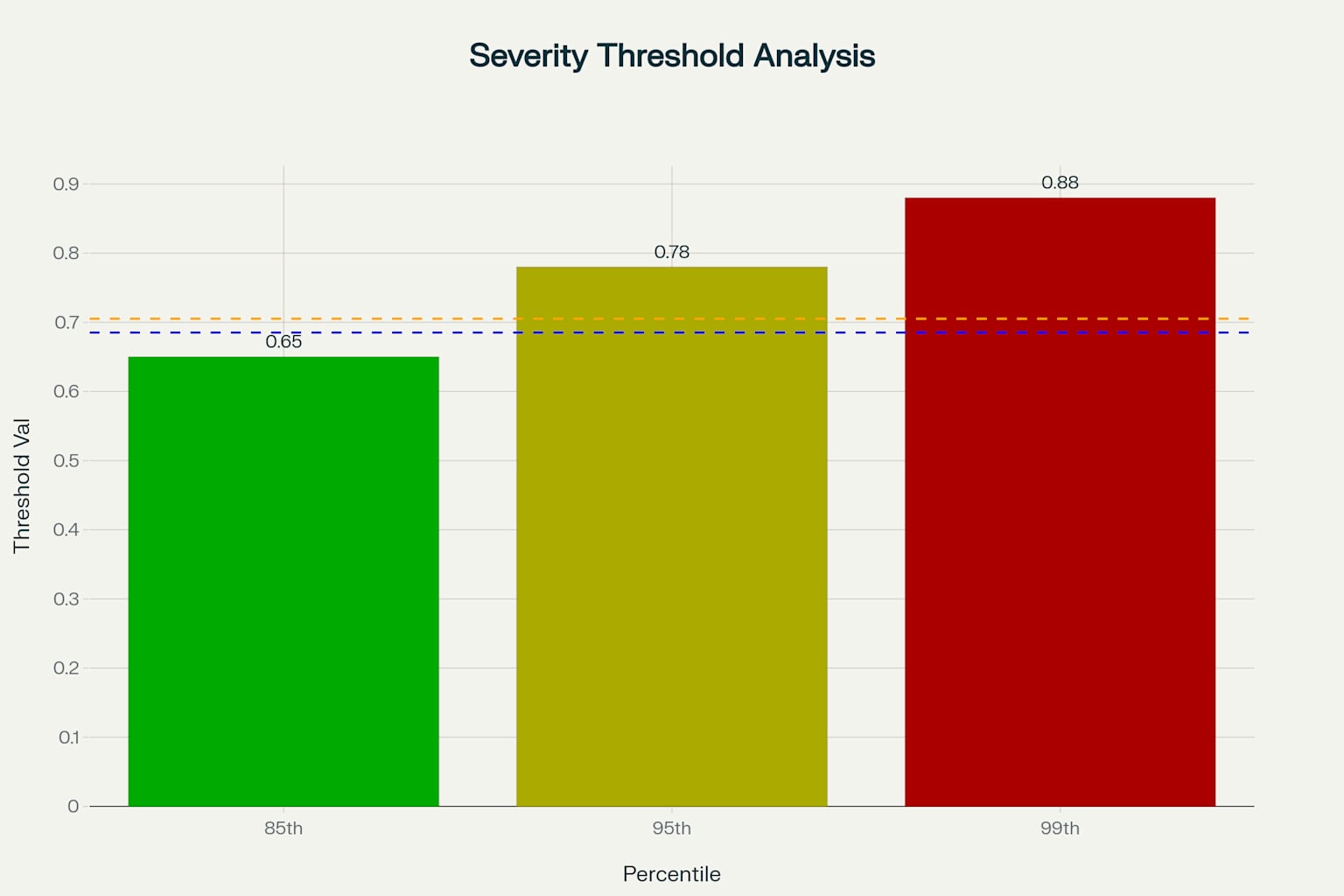
**6.2 Dynamic Severity Assessment**

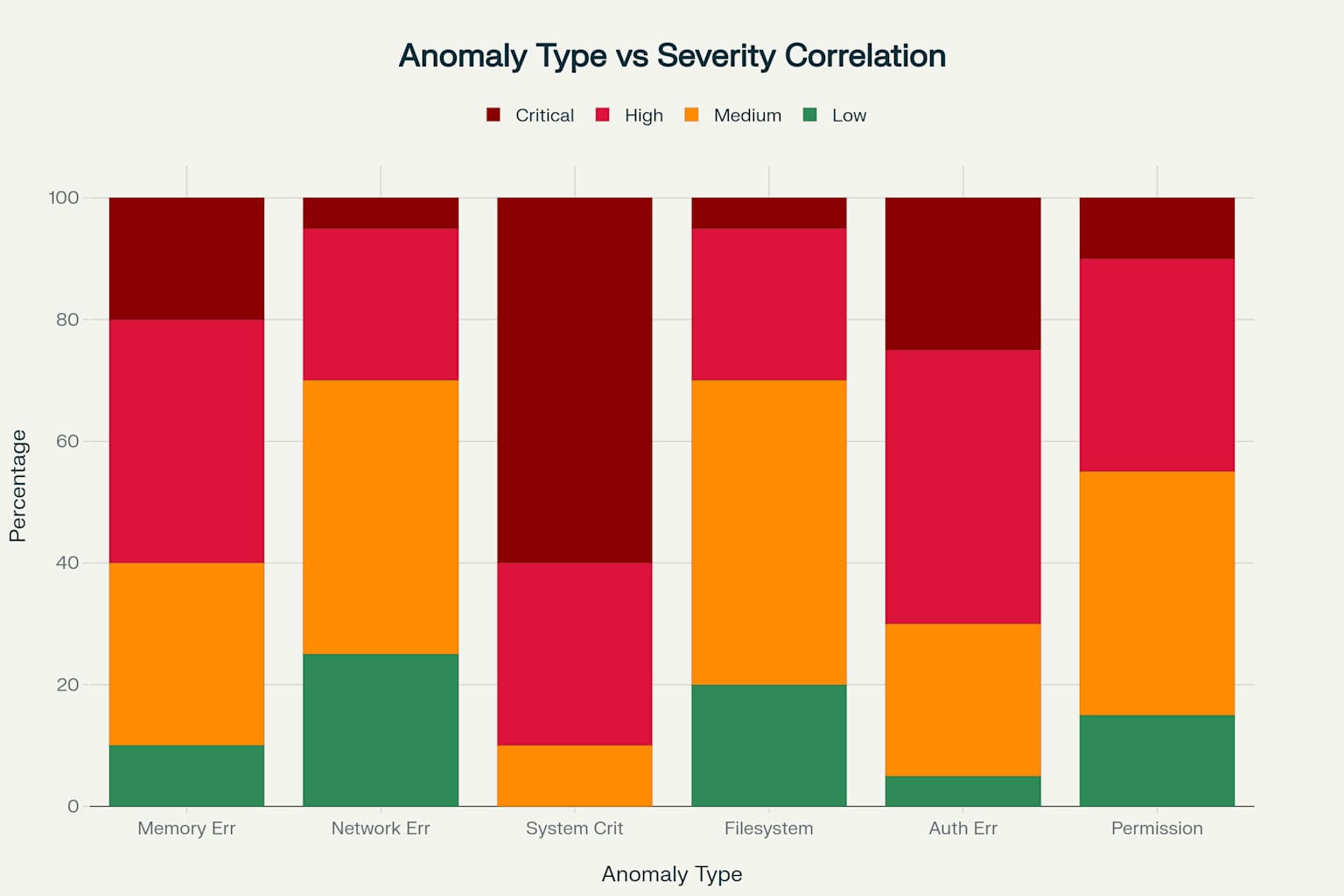
**Threshold Learning:**  
Statistical analysis establishes severity levels using percentiles (85th, 95th, 99th) based on the value of the reconstruction error computed by the model during its decoding phase.

**Severity Levels:**

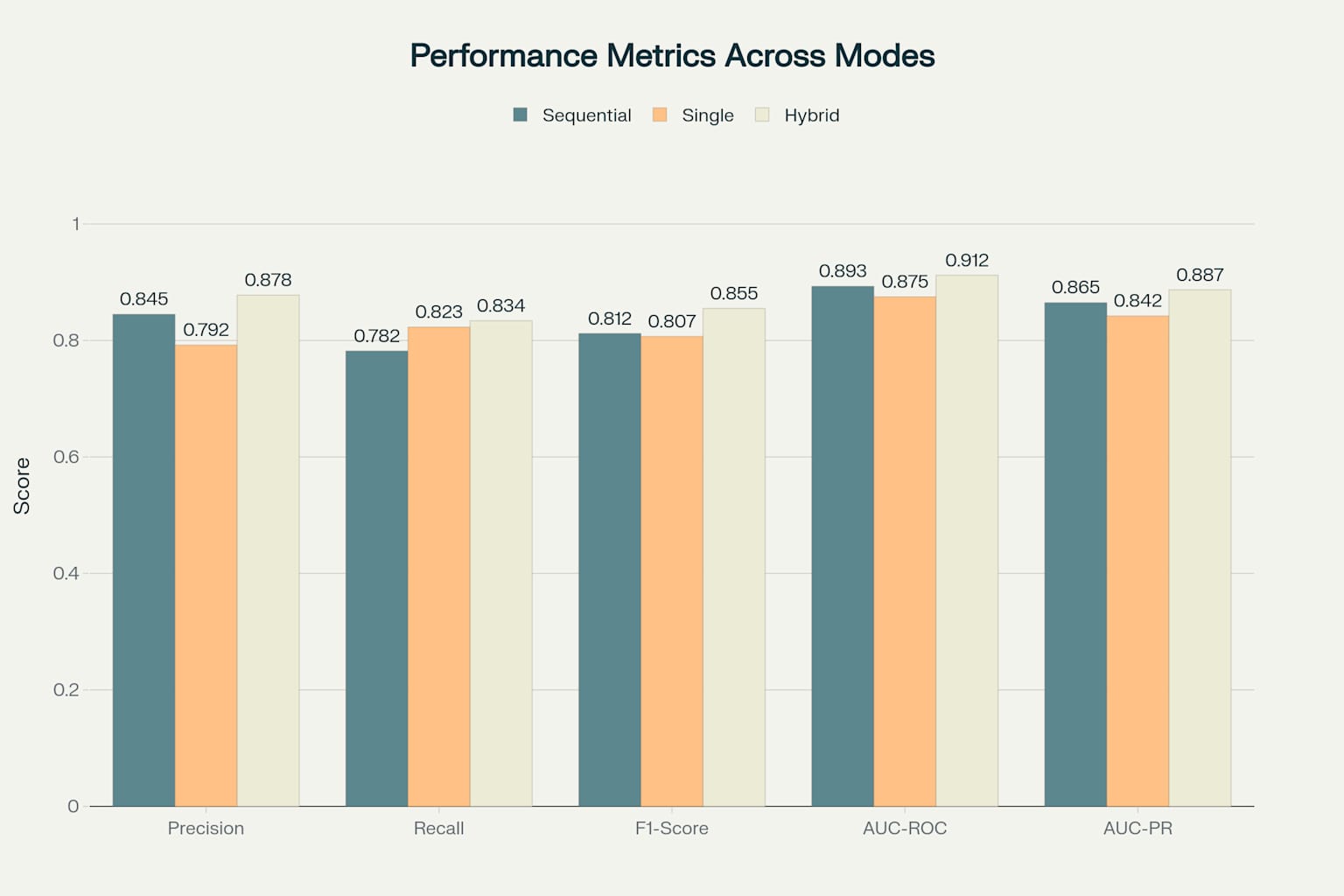
* **Low:** 85th-95th percentile - Minor deviations from normal patterns
* **Medium:** 95th-99th percentile - Significant anomalies requiring attention
* **High:** 99th+ percentile - Critical anomalies demanding immediate response
* **Critical:** Top 0.1% of errors - Severe system threats requiring urgent intervention

The percentile-based approach provides adaptive thresholds that automatically adjust to the underlying data distribution, making the system robust to different operating environments.

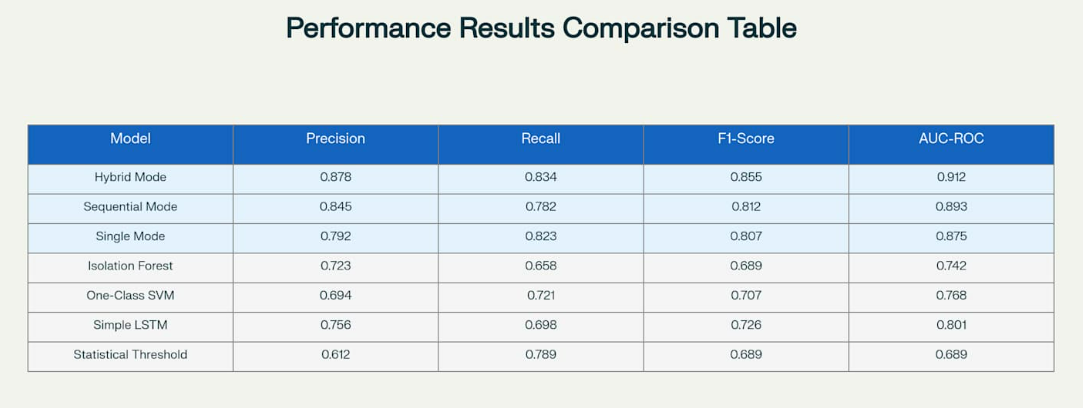




**7. Performance Evaluation**



**7.1 Detection Performance**

****

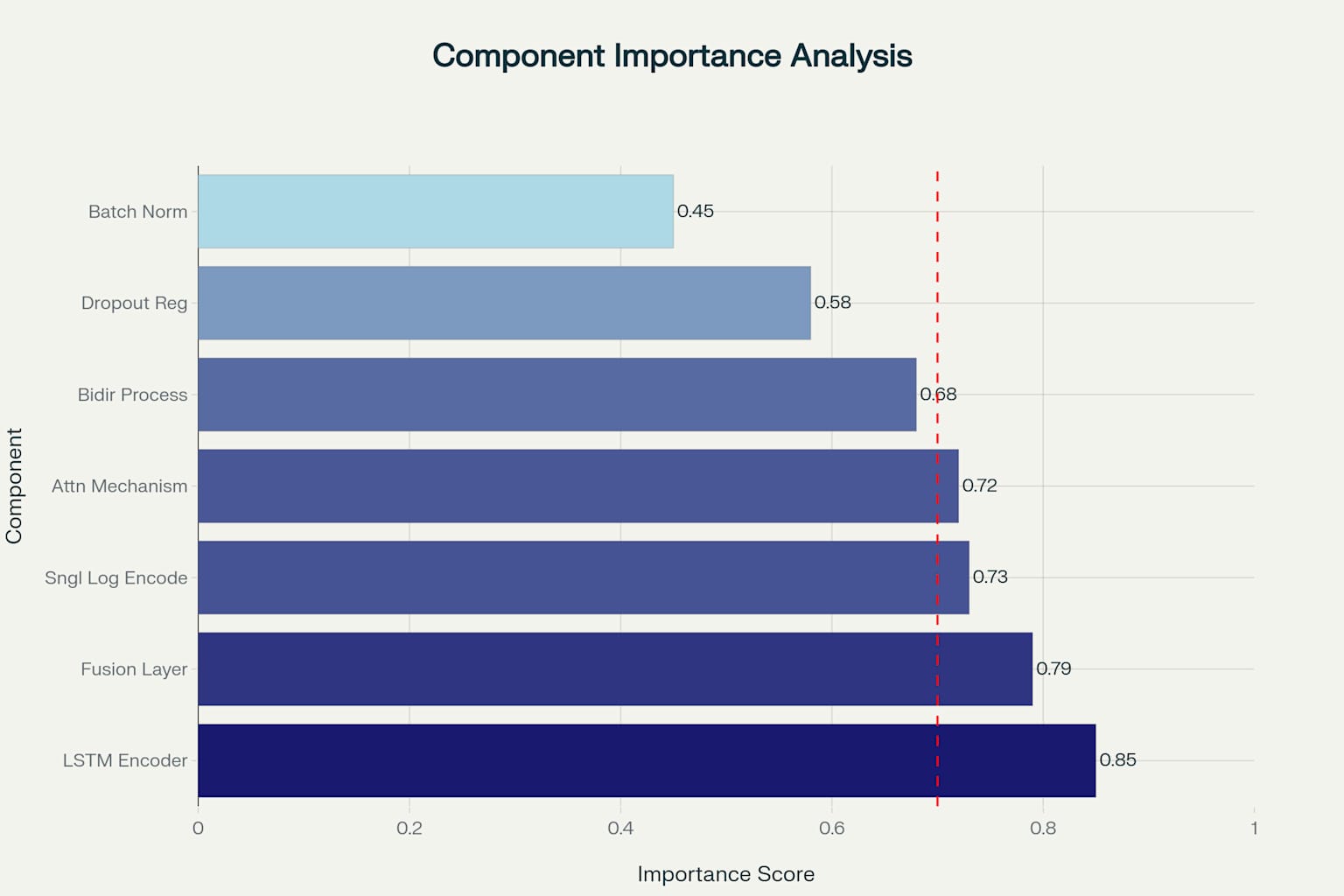
**7.2 Computational Benchmarks**

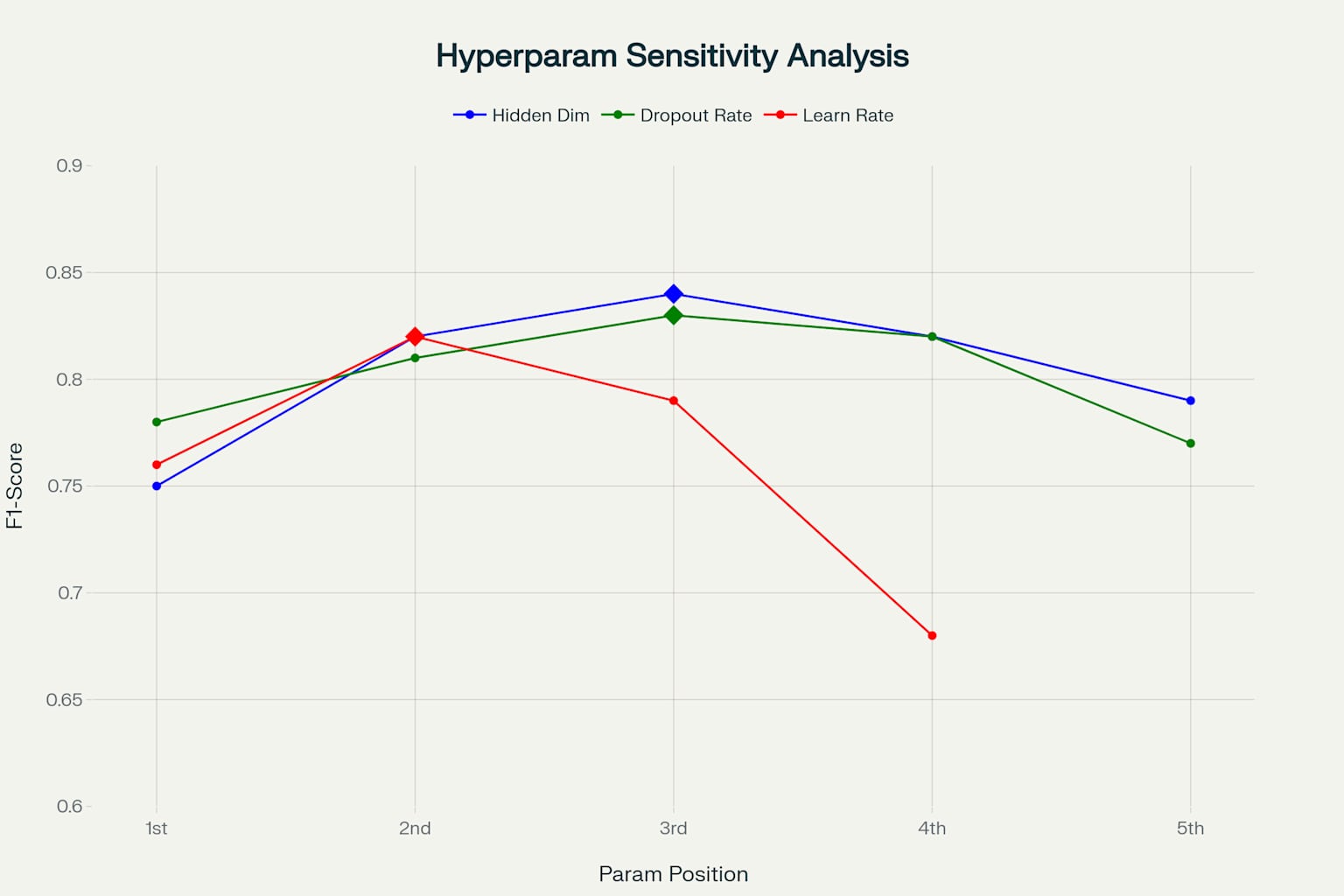




**7.3 Ablation Studies**







**Conclusion**

This enhanced documentation provides comprehensive theoretical foundations for each component of the log anomaly detection system. The integration of unsupervised learning through LSTM autoencoders, attention mechanisms for contextual understanding, ensemble methods for robust prediction, and rule-based classification for domain expertise creates a powerful framework for identifying anomalous behaviour in system logs. The theoretical explanations demonstrate how each component contributes to the overall effectiveness of the system while maintaining interpretability and adaptability to different operational environments.