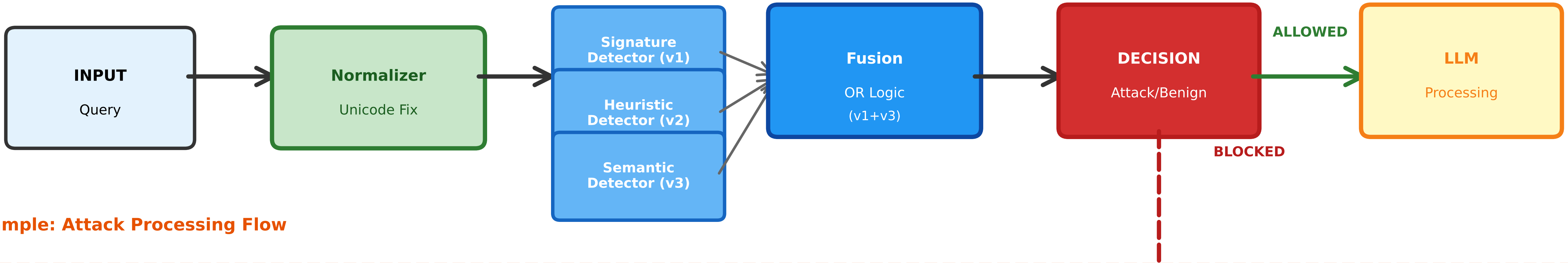
Prompt Injection Detection Pipeline Architecture

Input-Side Detection Before LLM Processing



INPUT
Query

Normalizer
Unicode Fix

Signature Detector (v1)

Heuristic Detector (v2)

Semantic Detector (v3)

Fusion
OR Logic
(v1+v3)

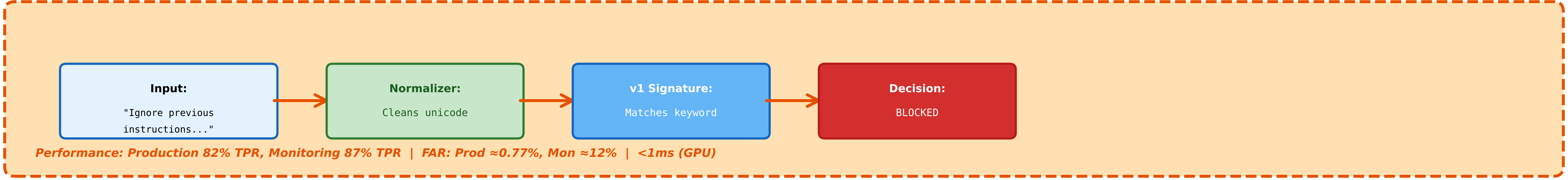
DECISION
Attack/Benign

ALLOWED

BLOCKED

LLM
Processing

Example: Attack Processing Flow



Input:
"Ignore previous instructions..."

Normalizer:
Cleans unicode

v1 Signature:
Matches keyword

Decision:
BLOCKED

Performance: Production 82% TPR, Monitoring 87% TPR | FAR: Prod ≈0.77%, Mon ≈12% | <1ms (GPU)

Production Configuration: Normalizer + v3

```
True Positive Rate (TPR): 82%
False Alarm Rate (FAR): 0.77%
Latency: <1ms per sample (GPU)
Complexity: ~1,200 lines
Deployment: Stateless
Dependencies: sentence-transformers, torch
```

Component Specifications

```
Signature Detector (v1):
  • 89% TPR, 0% FAR (P1)
  • Keyword matching
Semantic Detector (v3):
  • 82% TPR, 0% FAR (P1)
  • Pattern analysis
Fusion: OR Logic (v1+v3)
  • Monitoring: 87% TPR
```

Key Design Principles

1. INPUT-SIDE DETECTION: Attacks blocked BEFORE reaching the LLM
2. NORMALIZER FIRST: Unicode/homoglyph normalization ensures consistent detection
3. COMPLEMENTARY DETECTORS: v1 (signature) + v3 (semantic) catch different patterns
4. THRESHOLD-INVARIANT: Binary OR logic eliminates threshold tuning complexity
5. PRODUCTION-READY: <1ms latency with GPU acceleration, stateless architecture

Legend: □ Input □ Normalizer ■ Detector ■ Fusion ■ Decision □ LLM