Pressure Sensor Data Analysis: Can We Teach Computers to Detect Jumps?

Joshua Gao[1⋆], Alexander Wei[1]

Advisor: Larry Wilen[1]

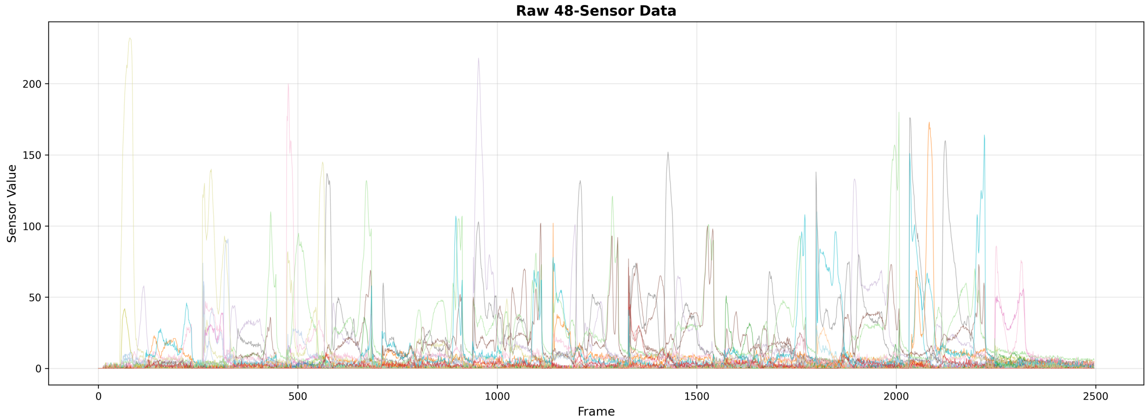
⋆ Submitted in partial fulfillment of the requirements for the Bachelor of Science degree

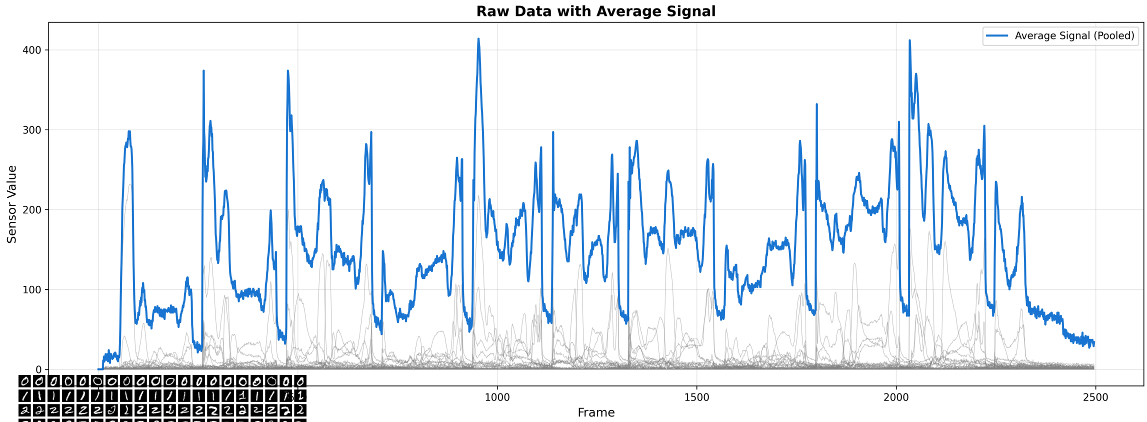
[1]School of Engineering and Applied Science, Yale University

Motivation

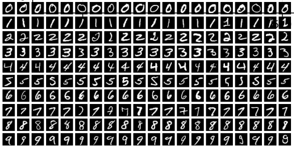
- Athletic training has become increasingly data-driven
- Three key steps: (1) data collection, (2) analysis, (3) translating insights
- **Challenge:** Can we teach computers to do step 2 of this process?
- **Raw Data:** Ground reaction forces collected from a 48-sensor pressure plate (50 Hz)

Raw 48-Sensor Data





Raw Data with Average Signal
Average Signal (Pooled)
Sensor Value
0
100
200
300
400
1000
1500
2000
2500
Frame



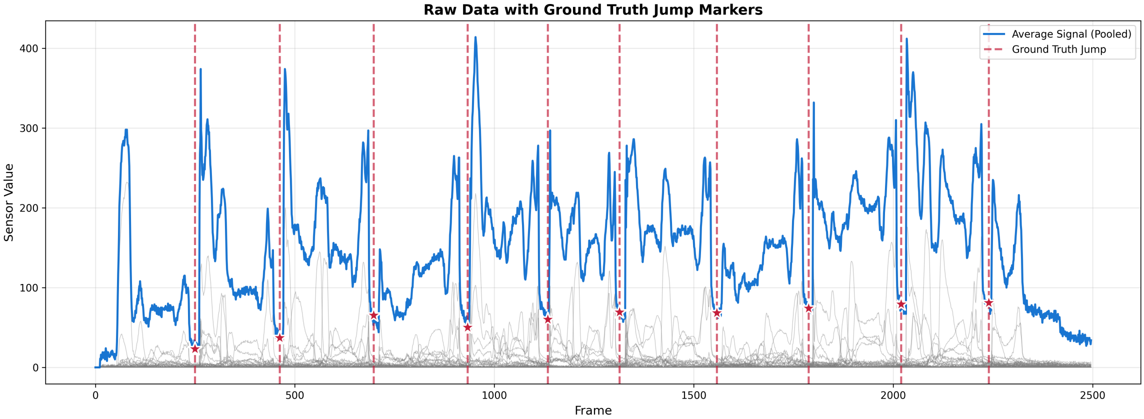
Human Intuition for Analyzing Jumps

Vertical jumps exhibit a characteristic signature:

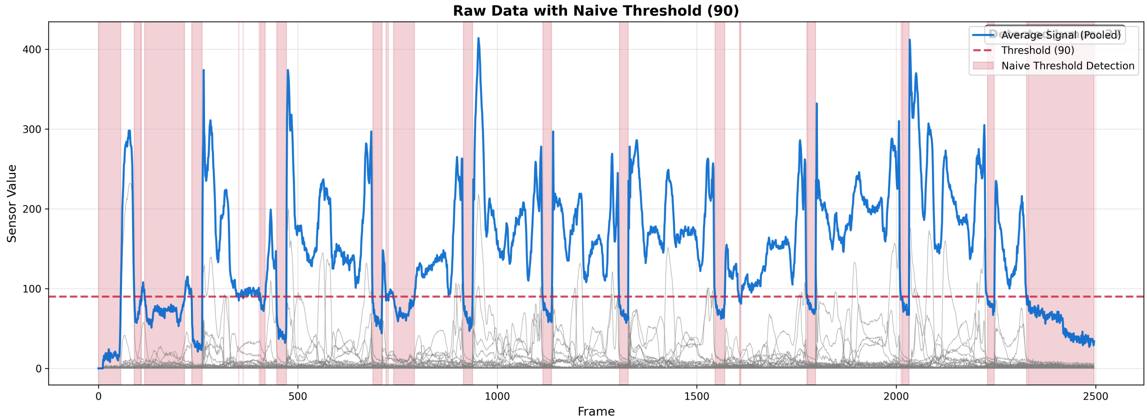
1. Force build-up during countermovement and propulsion
2. Rapid unloading at takeoff
3. Near-zero force while airborne
4. Rapid reloading at landing

Challenge: While patterns are visually obvious to humans, automated detection in noisy real-world conditions is difficult.

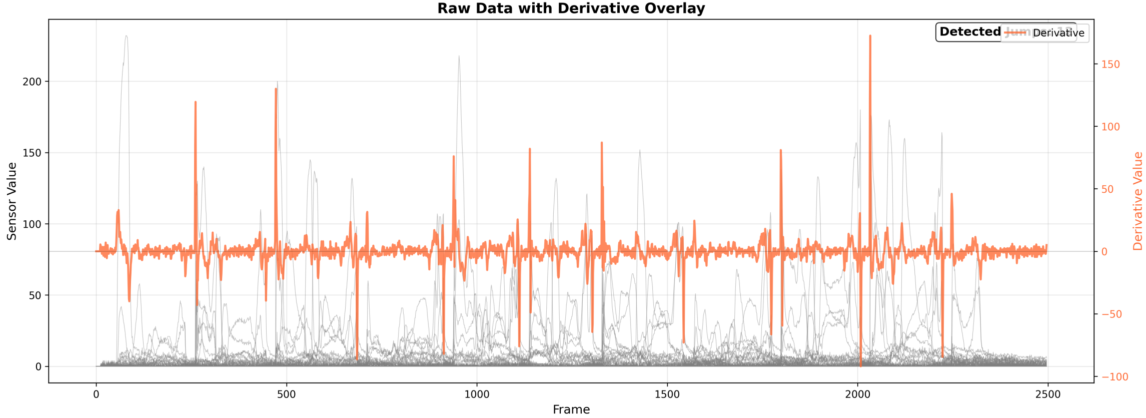
Raw Data with Ground Truth Jump Markers



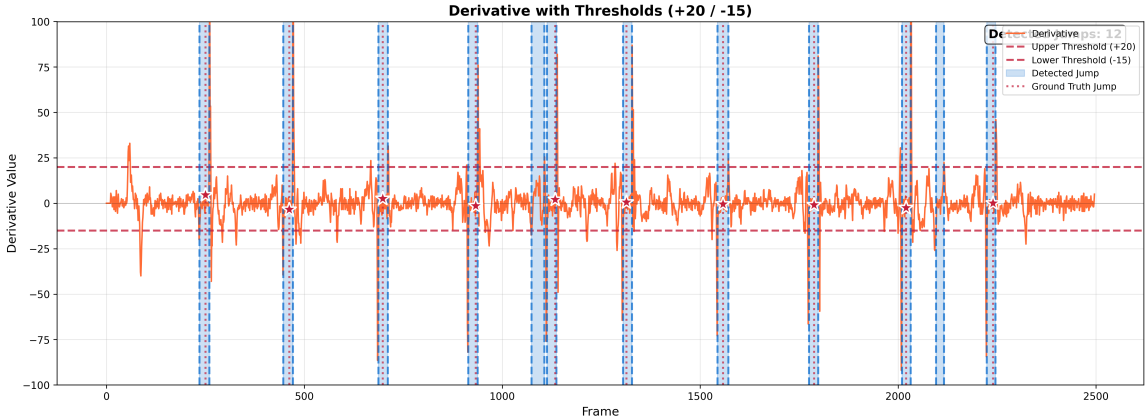
Naive Threshold Detection (Threshold = 90) (21 Jumps Detected!)



Raw Data with Derivative Overlay



Derivative-Based Detection (Thresholds: +20 / -15) (13 Jumps Detected)



Threshold & Derivative Algorithms

Threshold Algorithm:

- Identifies flight periods when force falls below threshold θ
- Binary mask: $m_{\text{threshold}}(t) = 1$ if $\bar{d}(t) < \theta$
- Filtered by physics constraints: $t_{\text{min}} \approx 0.2$ s, $t_{\text{max}} \approx 1.2$ s

Derivative Algorithm:

- Detects paired takeoff and landing events in derivative signal
- Upper threshold: $d'(t) > \theta_{\text{upper}}$ (landing)
- Lower threshold: $d'(t) < \theta_{\text{lower}}$ (takeoff)
- Validated by in-air threshold: $\bar{d}(t) < \theta_{\text{air}}$ during flight

Both algorithms operate in $O(T)$ time, enabling real-time mobile deployment.

All Algorithms & Parameters to Optimize

Algorithm	Brief Description	Parameters to Optimize
Threshold	Identifies flight periods when force falls below threshold. Binary mask filtered by physics constraints.	Threshold θ, derivative threshold δ, minimum flight time t_{min}, maximum flight time t_{max}
Derivative	Detects paired takeoff and landing events in derivative signal using upper/lower thresholds, validated by in-air threshold.	Upper threshold θ_{upper}, lower threshold θ_{lower}, in-air threshold θ_{air}, minimum flight time t_{min}, maximum flight time t_{max}
Correlation	Template matching on derivative signal using configurable buffer encoding jump signature (preparation, transition, jump phases).	Buffer size B, negative frames w_{neg}, zero frames w_{zero}, positive frames w_{pos}, correlation threshold θ_{corr}
Hybrid	Combines threshold-based takeoff detection with derivative-based landing detection. Pairs takeoff events with landing events.	Takeoff threshold θ_{takeoff}, landing derivative threshold θ_{landing}, in-air threshold θ_{air}, minimum flight time t_{min}, maximum flight time t_{max}
Ensemble	Combines multiple algorithms (threshold, derivative, correlation, hybrid) through weighted voting on binary condition indicators.	Weights for 11 condition signals, score threshold, minimum flight time t_{min}, maximum flight time t_{max}

Scoring Each Algorithm: Loss Function

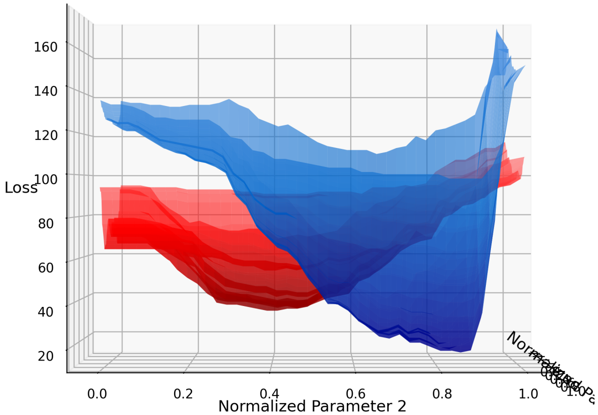
Loss Function:

$$L(\theta) = \text{FP}(\theta) + \text{FN}(\theta)$$

- **False Positives (FP):** Detected jumps without corresponding ground truth markers
- **False Negatives (FN):** Ground truth markers not captured by any detected jump

Manual Tagging: All jump events manually annotated in dataset

Means of Optimization: Grid Search Initial Parameters



Two-Dimensional Grid Search: Threshold: $\theta \in [90, 300]$, $\delta \in [0, 100]$ (40 values each); Derivative: $\theta_{\text{upper}} \in [0, 100]$, $\theta_{\text{lower}} \in [-100, 0]$ (40 values each)

Langevin Monte Carlo Sampling Optimization

Methodology: Stochastic optimization technique inspired by Langevin dynamics for efficient exploration of high-dimensional parameter spaces.

- **Metropolis-Hastings acceptance criterion:** Accepts parameter changes that decrease loss, or probabilistically accepts increases based on temperature
- **Temperature schedule:** Initial temperature 10.0 with exponential decay rate 0.9995 per iteration
- **Escape from local minima:** Temperature schedule enables exploration beyond immediate improvements
- **Comprehensive optimization:** 10,000 iterations for multi-parameter optimization across all algorithms

Advantages over grid search:

- Computationally feasible for high-dimensional spaces (e.g., 5+ parameters)
- Efficiently explores parameter space without exhaustive enumeration
- Finds optimal configurations that minimize loss function $L(\theta) = \mathrm{FP}(\theta) + \mathrm{FN}(\theta)$

Results

Algorithm	Loss	Parameters Optimized
Threshold	107	θ, δ, t_{min}, t_{max}
Derivative	**28**	θ_{upper}, θ_{lower}, θ_{air}, t_{min}, t_{max}
Correlation	229	B, w_{neg}, w_{zero}, w_{pos}, θ_{corr}
Hybrid	72	θ_{takeoff}, θ_{landing}, θ_{air}, t_{min}, t_{max}
Ensemble	30	Weights (11), score threshold, t_{min}, t_{max}
Landing Derivative	34	θ_{landing}, offset, window, θ_{air}, t_{min}, t_{max}

Final Model Selection & Deployment

Optimized Parameters:

- $\theta_{\text{upper}} = 17.71$, $\theta_{\text{lower}} = -15.19$, $\theta_{\text{air}} = 183.04$
- $t_{\text{min}} = 0.19$ s, $t_{\text{max}} = 0.60$ s

Implementation:

- JavaScript implementation for real-time processing
- Deployed in **Vault One** mobile application
- $O(T)$ complexity enables real-time mobile deployment
- Used by trainers and athletes for immediate feedback

Thank You