Machine Learning for Gait Event Detection & Pain Anamnesis Classification

Hassan Rasheed, Jaeyeop Chung
Advisors: Lucas Heitele, Pietro Fantini
Supervisor: Prof. Dr. Joachim Henkel
Application Project with Eversion Tech

Background: Gait Event Detection

Gait Events: Critical moments in the walking cycle (e.g. heel strike, toe-off) used in clinical gait analysis and prosthetics calibration

Traditional Methods: Typically rely on force plate thresholds or foot-switch sensors to detect events [1]. These rule-based approaches require careful sensor placement and tuning.

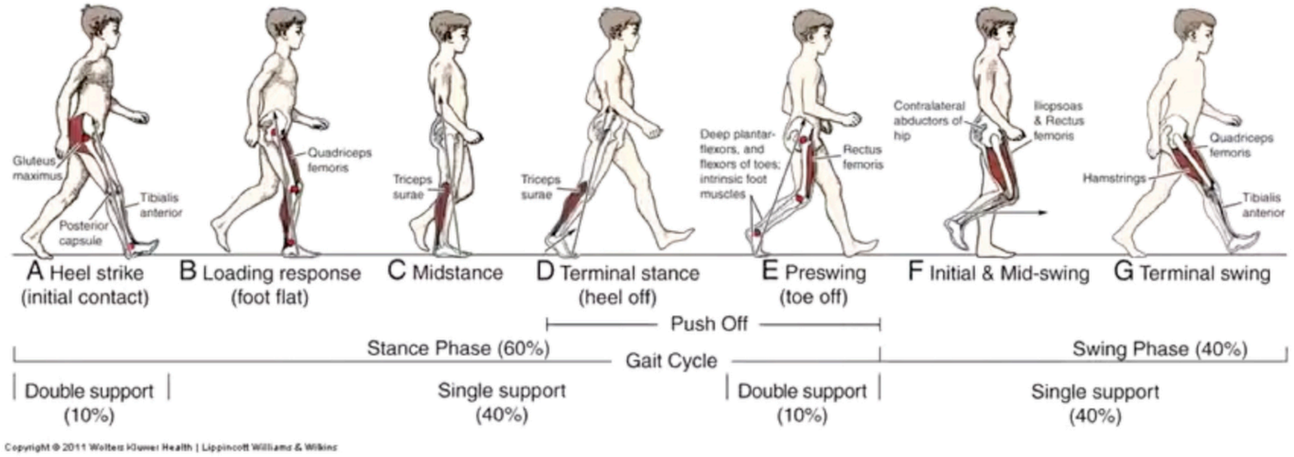


Figure: Gait event cycle [2]

[1] PLOS ONE – Chen & Martin (2025), Automated gait event detection for exoskeleton-assisted walking using a long short-term memory model with ground reaction force and heel marker data.
[2] Image source : https://www.youtube.com/watch?v=XqtAzJ2fNE4&list=PLFPcyJt53iSFlVWsiFuaqRCHx7Ex7axGp

Background: Gait Event Detection

Challenges: Threshold methods can fail under variable conditions (e.g. different speeds, devices) [1]. They need prior knowledge of sensor orientation and often miss subtle variations [2].

Opportunity for ML: Data-driven models (e.g. neural networks) learn gait patterns directly from inertial measurement unit (IMU) signals, handling variability better than fixed rules [1][3]. Recent studies show LSTM and deep learning models can exceed 90% accuracy in gait phase detection [4].

[1] PLOS ONE – Chen & Martin (2025), Automated gait event detection for exoskeleton-assisted walking using a long short-term memory model with ground reaction force and heel marker data.
[2] Romijnders R et al. (2022), A Deep Learning Approach for Gait Event Detection from a Single Shank-Worn IMU: Validation in Healthy and Neurological Cohorts
[3] Vu et al. (2020), A Review of Gait Phase Detection Algorithms for Lower Limb Prostheses
[4] Zhen et al (2019), Walking Gait Phase Detection Based on Acceleration Signals Using LSTM-DNN Algorithm

Background: Correlation of Gait Events and Pain Anamnesis

Gait event detection accurately identifies phases in walking patterns (e.g., heel strike, toe-off), while pain anamnesis classification determines the presence and severity of pain in various body regions.

Pain Anamnesis is patient's pain history recorded via questionnaires – often pain levels (0–5 or 0–10 scale) for various body regions (foot, arm, back, etc.)

By correlating detected gait anomalies or variations with specific pain areas, clinicians can better understand how movement patterns reflect or influence pain, enabling targeted therapeutic interventions.

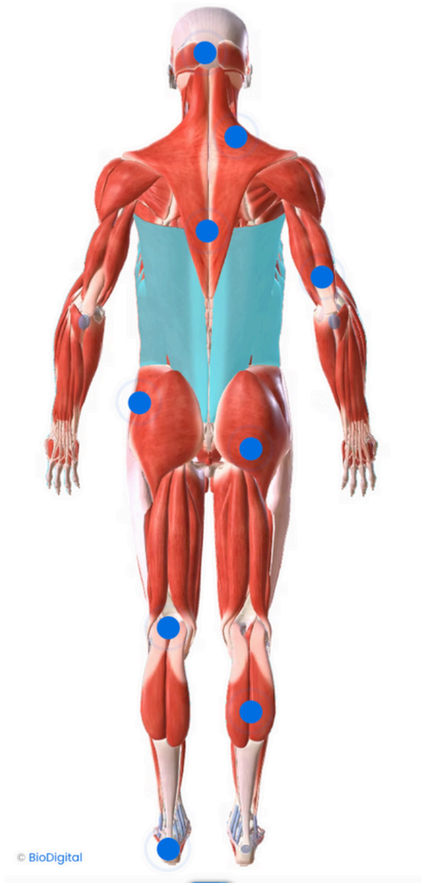


Figure: Pain anamnesis example from Eversion

Background: Pain Anamnesis Classification

Legacy Approach: Traditionally, clinicians interpret these pain scores qualitatively or use simple scoring. Automatic classification has been limited – often treating each pain site separately or ignoring severity due to complexity.

Challenges: Pain ratings on Likert/NRS scales are ordinal (ranked) [1]. Standard models either bin them into binary presence or treat them as numeric, risking loss of information or improper assumptions.

ML Advantages: Multi-output neural networks can handle many pain indicators at once (multi-label), uncovering hidden correlations. Ordinal-aware ML can utilize the rank information in pain severity, improving predictive insight beyond binary classification.

[1] Kim TK (2017), Practical statistics in pain research

Part 1: Gait Event Detection

Methodology - Two approaches we used

Using LSTMs: Effective at modeling temporal patterns in IMU sensor sequences, suitable for complex, dynamic gait event detection.

Using XGBoost: Excellent for structured data, providing strong predictive power, interpretability, and faster inference without sequence modeling complexity.

Methodology: Gait Event Detection (Data)

Sensor Data: 6-axis IMU readings (3-axis accelerometer + 3-axis gyroscope) from foot during walking. Data is segmented into short time-series windows (e.g. 2–3 second sequences) capturing at least one gait cycle

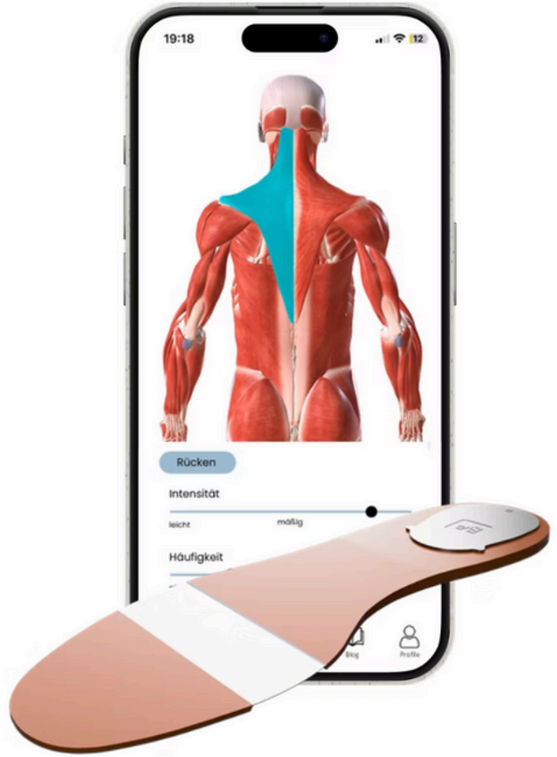


Figure: Eversion insole for gait analysis and app for anamnesis

Methodology: LSTM for Gait Event Detection

Why LSTM: Unlike static classifiers, LSTMs maintain an internal state, making them adept at detecting events defined by a sequence of signal patterns rather than any single timestamp.

This compact network learns to differentiate phases like heel strike vs toe off by the temporal pattern in IMU data, replacing the need for manual threshold logic.

Prior work also validates LSTM efficacy for gait event detection under varying speeds [1]

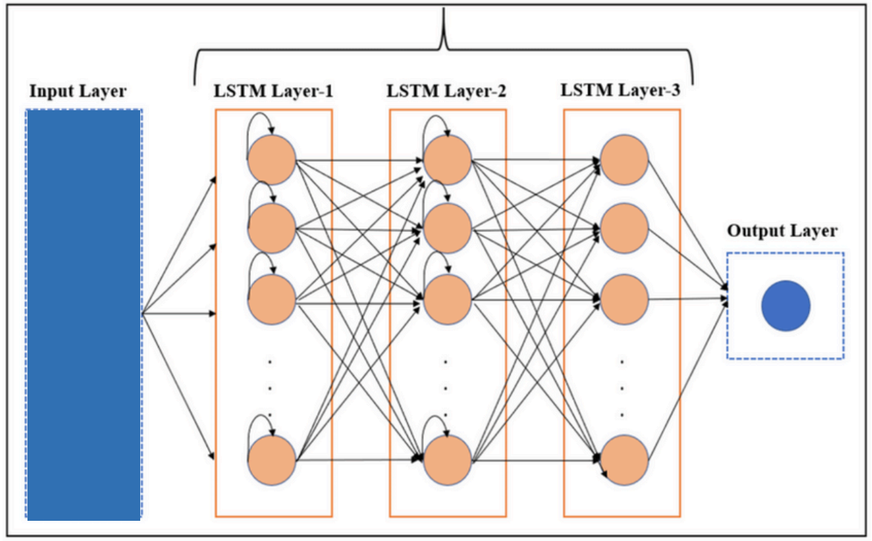


Figure: LSTM Architecture

[1] Zhen et al (2019), Walking Gait Phase Detection Based on Acceleration Signals Using LSTM-DNN Algorithm

Methodology: LSTM-Based Gait Event Detection

Model Architecture: Long Short-Term Memory (LSTM) recurrent neural network tailored for sequence classification. We use a **3-layer LSTM (128 hidden units each)** with a final dense layer to output **one of 5 classes (no event, heel strike, foot flat, heel off, toe off)**.

- The LSTM cells learn temporal dependencies in the sensor signals, effectively recognizing the distinctive acceleration/gyro patterns at each gait event
- Regularization: Dropout applied to LSTM layers to prevent overfitting. Data normalized (MinMaxScaler) so all features on comparable scale.

Training: Supervised learning with labeled gait phases.

- Used **~70%** data for training, **15%** validation, **15%** test.
- Optimizer Adam with learning rate **0.001**.
- Model trained for **~50** epochs until validation accuracy converged **(~91%)**

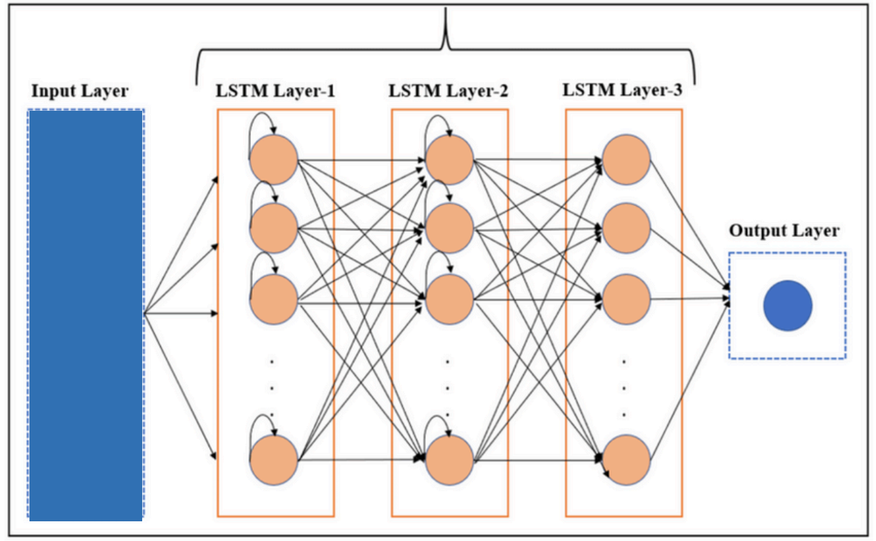


Figure: LSTM Architecture

Results: LSTM-Based Gait Event Detection

Overall Accuracy: The LSTM model achieves **~91% accuracy** in classifying gait events on the test set.

This is a substantial improvement over simple threshold methods (**often 84–90% at best** for multiple phases [1] and on par with state-of-the-art ML models [2].

Class-wise Performance: The model performs exceptionally on the dominant gait phases: for Foot Flat, Toe Off, **precision and recall are ~0.90–0.97 (F1 ≈ 0.92–0.96)**. Heel Off is also well-captured **(precision 0.82, recall 0.77)**.

However, infrequent events like Heel Strike were often missed (0 instances correctly predicted in test) leading to 0% recall for that class.

[1] Vu et al. (2020), A Review of Gait Phase Detection Algorithms for Lower Limb Prostheses
[2] Zhen et al. (2019), Walking Gait Phase Detection Based on Acceleration Signals Using LSTM-DNN Algorithm

Results: LSTM-Based Gait Event Detection

The table below (classification report) summarizes the performance per class

Gait Event	Precision	Recall	F1-Score	Support
No Event	0.84	0.21	0.34	265
Heel Strike	0.00	0.00	0.00	33
Foot Flat	0.90	0.94	0.92	22,740
Heel Off	0.82	0.77	0.80	14,536
Toe Off	0.96	0.97	0.96	28,474
Accuracy	-	-	**0.91**	66,048

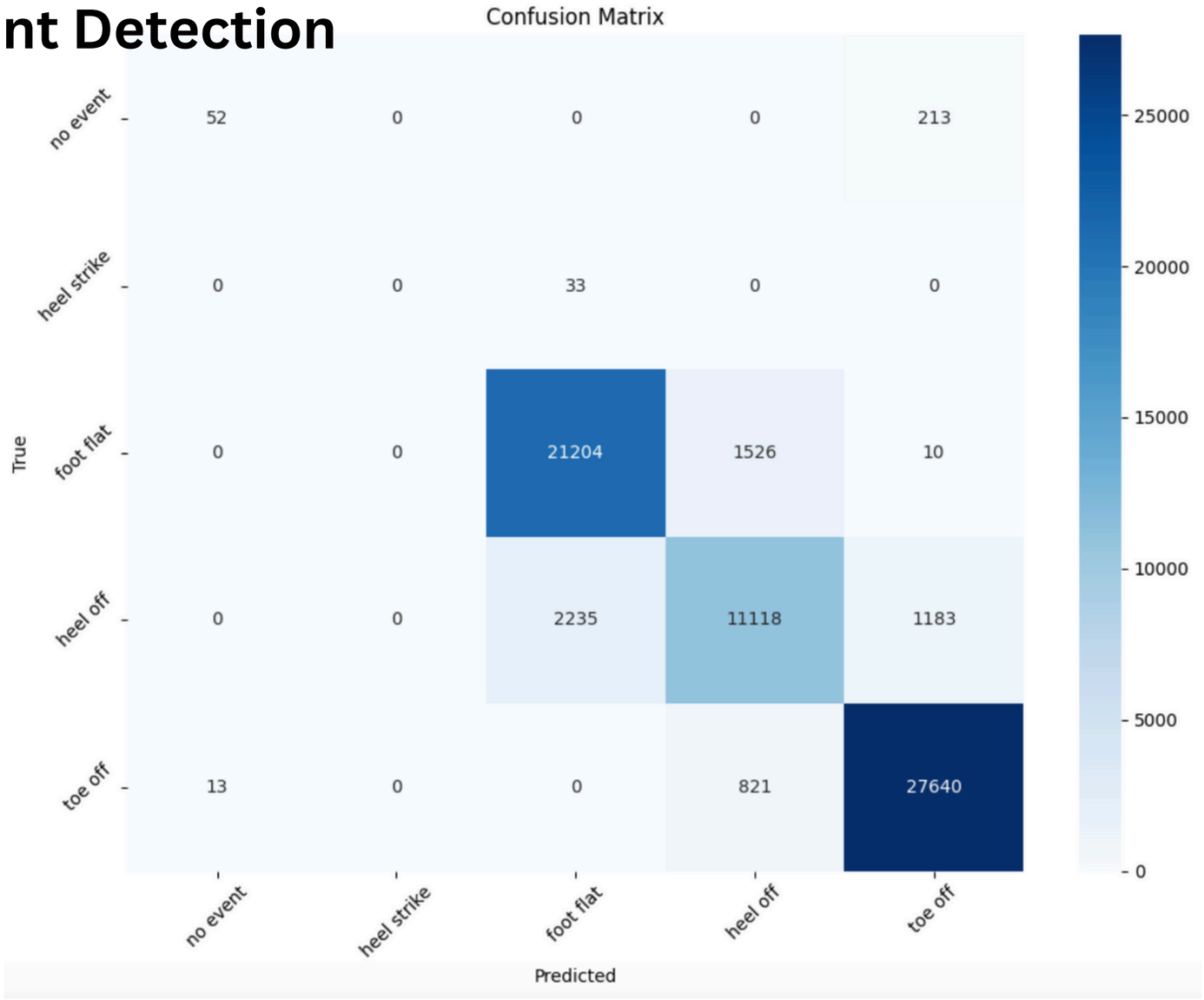
F1-score is the harmonic mean of precision and recall. 2*(precision*recall)/precision+recall

Class imbalance explains the disparity: “Heel strike” events were rare (only 33 instances vs tens of thousands of others) making it hard for the model to learn them.

The **weighted average F1 is 0.91**, reflecting that for the vast majority of time points in gait, the model predicts correctly.

Results: LSTM-Based Gait Event Detection

The confusion matrix shows the number of true positives against the predictions by the model.



Results: LSTM-Based Gait Event Detection

Takeaway: The LSTM-based approach robustly detects gait events in IMU data, automating what used to require dedicated hardware or manual threshold tuning. This accuracy **(~91%)** is on par with other state-of-the-art deep learning approaches **(e.g. CNN-BiGRU achieving ~93% [1])**.

The model's only notable weakness is in under-represented events, suggesting that providing more balanced training data or using data augmentation could further improve performance for those cases.

Overall, this demonstrates the feasibility of deploying RNN models in wearable gait analysis systems for real-time event detection.

[1] Arshad et al. (2022), Gait Events Prediction Using Hybrid CNN-RNN-Based Deep Learning Models through a Single Waist-Worn Wearable Sensor

Gait Event Detection - XGBoost : Data (1)

Data Statistics

- 6-axis IMU sensor data (3-axis accelerometer + 3-axis gyrometer)
- Measured at the rate of 200Hz (each internval = 5ms)
- Data are collected per each step (one gait phase)
  - Heel Strike, Foot Flat, Heel Off, Toe Off
- Data Statistics
  - 2324 unique collection ids
  - 8 columns (6-axis sensor data + collection id + gait phase)

	collection_id	gyroscope_x	gyroscope_y	gyroscope_z	accelerometer_x	accelerometer_y	accelerometer_z	phase
0	vh92aaJeQLxDazem2hN5_6_L	168.630	-158.620	-113.820	-1.026752	2.276032	-8.625400	1
1	vh92aaJeQLxDazem2hN5_6_L	-71.890	-272.580	-129.920	-2.516616	-1.586000	-2.127680	1
2	vh92aaJeQLxDazem2hN5_6_L	-129.850	-334.530	-114.100	-0.890112	-0.234240	0.258640	1
3	vh92aaJeQLxDazem2hN5_6_L	-206.850	-334.600	-111.300	-0.842288	0.425048	-1.624064	1
4	vh92aaJeQLxDazem2hN5_6_L	-223.650	-354.620	-105.700	0.108824	0.293288	-1.969568	1
...	...	...	...	...	...	...	...	...
508441	rbk3m0CFK4mQVt2ot6T2_6_R	-70.140	14.910	-201.460	4.293912	-0.462136	-0.983564	4
508442	rbk3m0CFK4mQVt2ot6T2_6_R	-39.970	-40.565	-188.160	4.011116	-0.584136	-1.016504	4
508443	rbk3m0CFK4mQVt2ot6T2_6_R	-10.885	-91.315	-168.525	3.681228	-0.622200	-1.105076	4
508444	rbk3m0CFK4mQVt2ot6T2_6_R	12.250	-132.860	-143.360	3.318644	-0.628300	-1.234152	4
508445	rbk3m0CFK4mQVt2ot6T2_6_R	27.790	-168.490	-110.845	2.846260	-1.909544	-1.917596	4

508446 rows × 8 columns

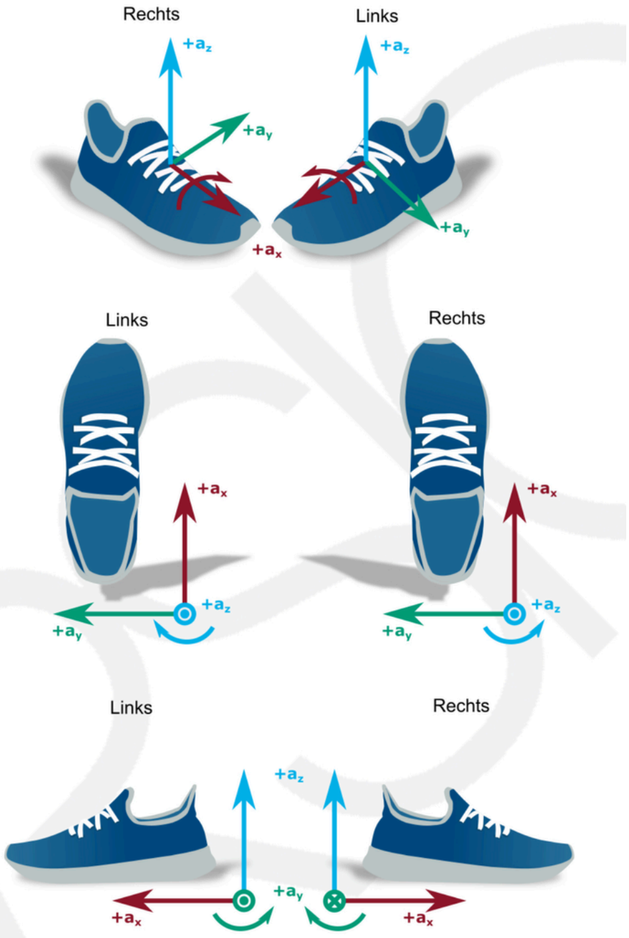


Figure 1: Foot Frame

Gait Event Detection - XGBoost : Data (2)

Measuring Autocorrelation and Identifying the Optimal Lag Size

- Autocorrelation : correlation of an observation at a given time with its lagged observations.
- Autocorrelation Function(**ACF**) is commonly used to mesaure autocorrelation of time series
  - The optimal lag is determined at the lag where ACF crosses confidential interval; *cutoff lag*

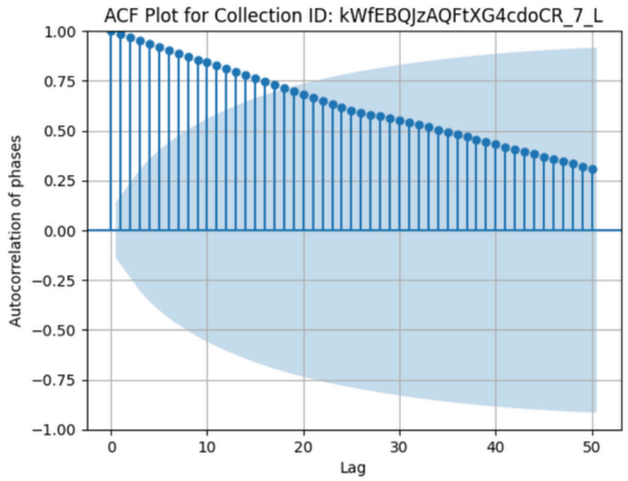
- Cutoff lags are calculated for all collection data(*2324 collection ids*) and used the average of these cutoff lags as the optimal lag size to construct lagged features.
  - The average of cutoff lags : 19

ACF values are calculated as follows

- $ACF(k) = \frac{\sum_{t=k+1}^{n}(y_t-\bar{y})(y_{t-k}-\bar{y})}{\sum_{t=1}^{n}(y_t-\bar{y})^2}$
  - $\bar{y} = \frac{1}{n}\sum_{t=1}^{n} y_t$,
  - $\mathrm{Var}(Y_t) = \frac{1}{n}\sum_{t=1}^{n}(y_t-\bar{y})^2$, $(k = \mathrm{lag}, t = \mathrm{time\ step})$
  - $\mathrm{Cov}(Y_t, Y_{t-k}) = \frac{1}{n-k}\sum_{t=k+1}^{n}(y_t-\bar{y})(y_{t-k}-\bar{y})$

Confidential Interval(Blue Region) is calculated as follows

- $SE = \sqrt{\frac{1+2\sum_{j=1}^{k-1}\rho(j)^2}{n}}$, ($\rho(j)$: ACF values at lag j, k : lag, n : number of data)
- $CI = \pm 1.96 \cdot SE$



Autocorrelation Function for one collection
(Confidential Interval at a lag k : blue region)

Gait Event Detection - XGBoost : Data (3)

Lagged Features

- Lagged features capture temproal dependencies and past influences to improve predictions
- For each sensor data (3-axis accelerometer, 3-axis gyrometer), 19 lagged features were constructed; *a total of 19 * 6 features*
- Lagged data statistics
  - 2324 unique collection ids
  - 116 columns (114 lagged features + collection_id + gait phas)

	gyroscope_x_lag1	gyroscope_x_lag2	gyroscope_x_lag3	gyroscope_x_lag4	gyroscope_x_lag5	gyroscope_x_lag6	gyroscope_x_lag7	gyroscope_x_lag8	gyroscope_x_lag9	gyroscope_x_lag10	...	accelerometer_z_lag12	a
0	0.573279	0.574568	0.573335	0.569859	0.565710	0.564196	0.560832	0.560327	0.565317	0.559094	...	0.480505	
1	0.568569	0.573279	0.574568	0.573335	0.569859	0.565710	0.564196	0.560832	0.560327	0.565317	...	0.474632	
2	0.565710	0.568569	0.573279	0.574568	0.573335	0.569859	0.565710	0.564196	0.560832	0.560327	...	0.469817	
3	0.565429	0.565710	0.568569	0.573279	0.574568	0.573335	0.569859	0.565710	0.564196	0.560832	...	0.467684	
4	0.564701	0.565429	0.565710	0.568569	0.573279	0.574568	0.573335	0.569859	0.565710	0.564196	...	0.467835	
...	...	...	...	...	...	...	...	...	...	...	...	...	
462479	0.483488	0.479031	0.491057	0.518334	0.553543	0.588164	0.609077	0.616170	0.610535	0.597668	...	0.467155	
462480	0.500392	0.483488	0.479031	0.491057	0.518334	0.553543	0.588164	0.609077	0.616170	0.610535	...	0.477719	
462481	0.524557	0.500392	0.483488	0.479031	0.491057	0.518334	0.553543	0.588164	0.609077	0.616170	...	0.485508	
462482	0.547853	0.524557	0.500392	0.483488	0.479031	0.491057	0.518334	0.553543	0.588164	0.609077	...	0.491900	
462483	0.566383	0.547853	0.524557	0.500392	0.483488	0.479031	0.491057	0.518334	0.553543	0.588164	...	0.497649	

462484 rows × 116 columns

Gait Event Detection - XGBoost : Methodology (1)

What is XGBoost?

XGBoost is a tree-based gradient boosting machine, exceptionally tailored for handling non-linear complex data with its resilience against multicollinearity.

- At each tree(iteration), XGBoost learns the temporal relation with the gait phase and the current residuals.
- The subsequent trees are built in a way that corrects the residuals of the previous tree, effectively refining its predictions.

This iterations enable the model to capture the temporal dependencies in the lagged features and to effectively predict the gait phases.

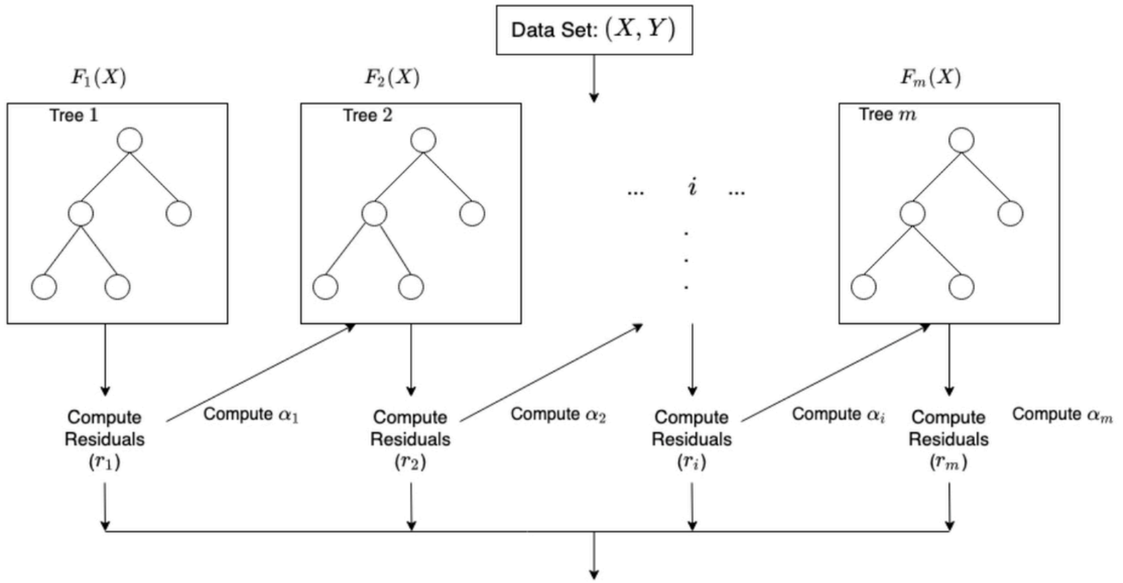


Figure: GBM architecture [1]

[1] https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html

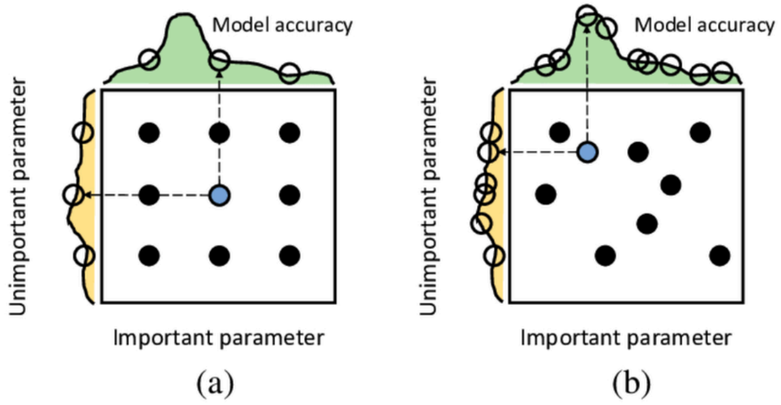
Gait Event Detection - XGBoost : Methodology (2)

XGBoost parameters

- What makes XGBoost powerful in handling complex tasks is its wide range of hyperparameters including those for regularization and GPU acceleration.
- To make the most out of XGBoost's performance, it is necessary to search for the optimal hyperparameters across a wide range of hyperparamer space using such techniques as cross validations at the cost of time.

Optuna

- **Optuna** is a tool to optimize hyperparameter optimization, which utilizes Bayesian method that models *the likelihood of the hyperparameter configurations*.
- This process is similar to cross validation but **Optuna**'s utilizing the likelihood of the hyperparameter space lets the model to reach the optimization goal more efficiently.



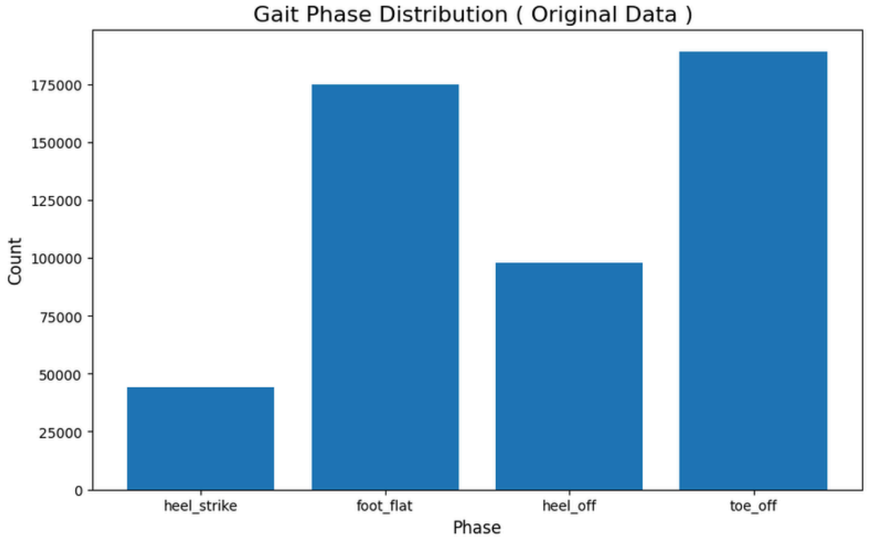

```
params = {
    'early_stopping_rounds': trial.suggest_int('early_stopping_rounds', 10, 200),
    'max_depth': trial.suggest_int('max_depth', 3, 10),
    'learning_rate': trial.suggest_float ('learning_rate', 0.01, 0.3),
    'n_estimators': trial.suggest_int('n_estimators', 100, 500),
    'min_child_weight': trial.suggest_int('min_child_weight', 1, 10),
    'subsample': trial.suggest_float('subsample', 0.5, 1.0),
    'colsample_bytree': trial.suggest_float('colsample_bytree', 0.5, 1.0),
    'gamma': trial.suggest_float('gamma', 0, 5),
}
```

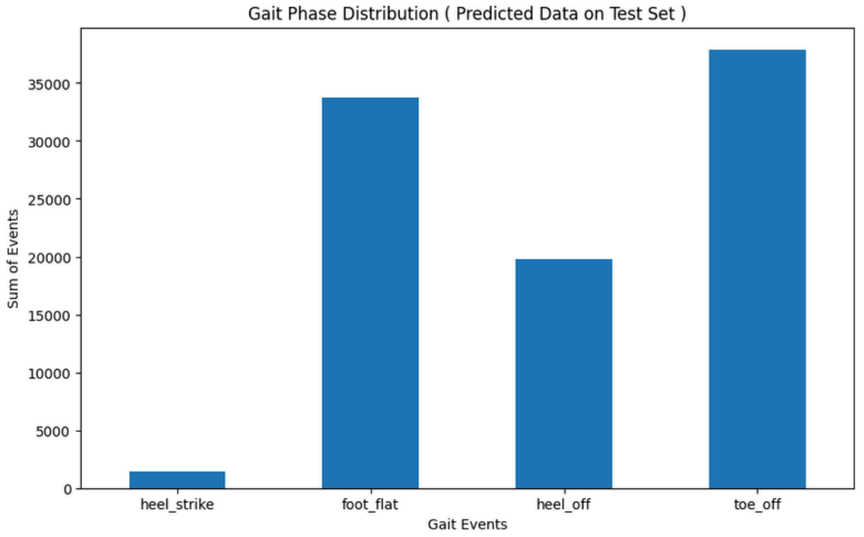
image source: https://jhtobigs.oopy.io/optuna_tutorial

Gait Event Detection - XGBoost : Result

- **Overall Accuracy** : **~92% accuracy** in classifying gait events on the test set
  - The trained model could well capture the distribution of the original data.
    - The **Kolmogorov-Smirnov (KS) test** between the original data and the predicted dataresulted in a **KS statistic of 0.8** and a **p-value of 0.7936**, indicating no statistically significant difference between the two distributions.

- Due to the class imbalances in *heel-strike* and *heel-off,* the model struggles to identify heel-strike classes, it may due to the model set the threshold for heel-strike too high.






```
test accuracy: 0.921006
                                                    
              precision    recall  f1-score   support

 heel strike       0.71      0.53      0.61      1417
   foot flat       0.91      0.94      0.93     33777
   heel off        0.86      0.81      0.83     19754
     toe off       0.97      0.97      0.97     37920

    accuracy                           0.92     92868
   macro avg       0.86      0.82      0.84     92868
weighted avg       0.92      0.92      0.92     92868
```

Part 2: Pain Anamnesis Classification

Approach: Pain Anamnesis Classification - Two Approaches

Dense Neural Network: Effectively models complex, non-linear relationships in multi-target pain classification tasks through deep feature extraction.

XGBoost: Offers strong predictive performance, interpretability, and efficient handling of structured biomechanical features for pain prediction.

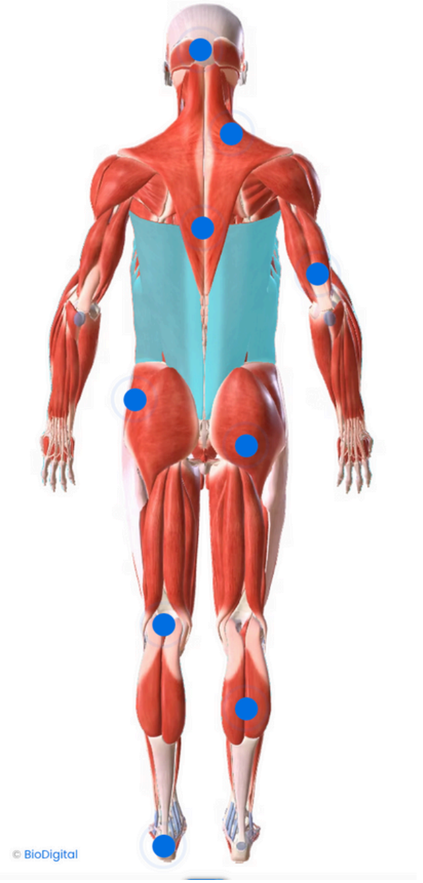


Figure: Pain anamnesis example from Eversion

Methodology: Pain Anamnesis Classification - Data

Input features include 7 biomechanical measurements **(e.g. left/right movement deviation, resting deviation, step length averages, and shoe size)** hypothesized to relate to pain outcomes.

Targets include **different pain areas of which 18 are binary and 24 ordinal**. The binary targets just store pain/no pain as boolean values whereas ordinal targets store the severity of the pain on a Likert scale from 0-5

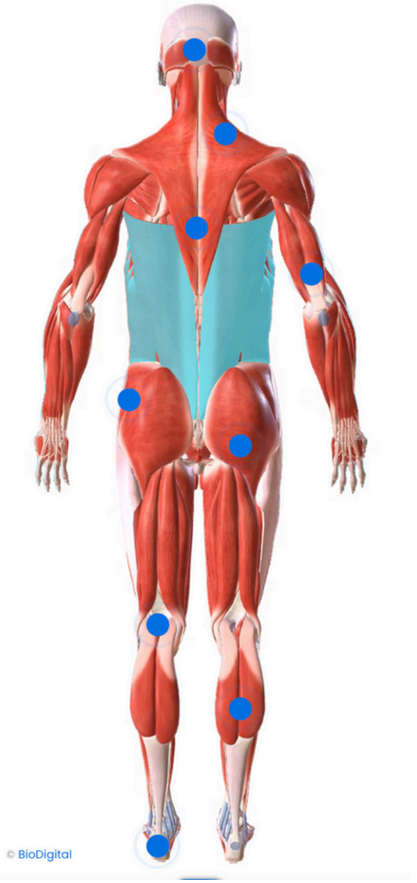


Figure: Pain anamnesis example from Eversion

Methodology: Pain Anamnesis Classification (Binary) - Dense NN

Data Preprocessing: Original pain dataset contains multiple Schmerz_* fields per body region, some as booleans **(pain Y/N)** and some as ordinal scores **(0 = no pain up to 5 = severe)**.

In the binary-only approach, all pain indicators were converted to **binary 0/1**: ordinal ratings were thresholded **(e.g. >3 = 1, else 0)** so that each region has a binary pain presence label

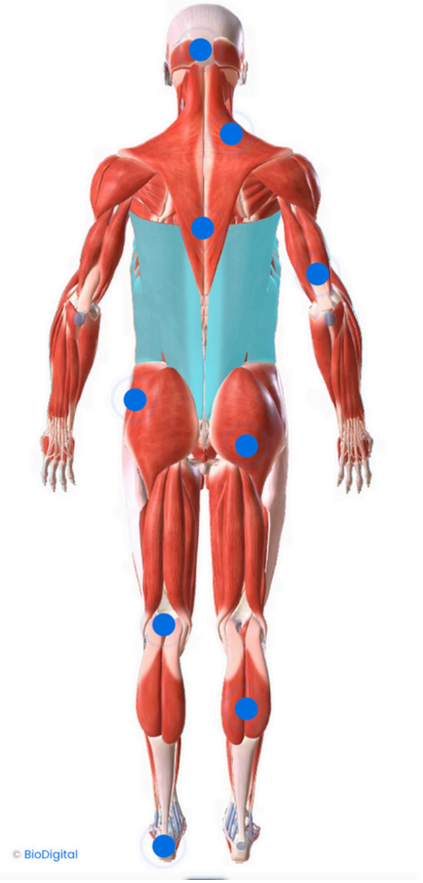


Figure: Pain anamnesis example from Eversion

Methodology: Pain Anamnesis Classification (Binary) - Dense NN

Rationale: This approach simplifies the problem by treating every pain site independently as “pain or no pain.”

It is straightforward and allows using standard classification techniques, but it discards the ordinal severity information (e.g. moderate vs severe pain both just treated as “pain”) and assumes independence among pain sites.

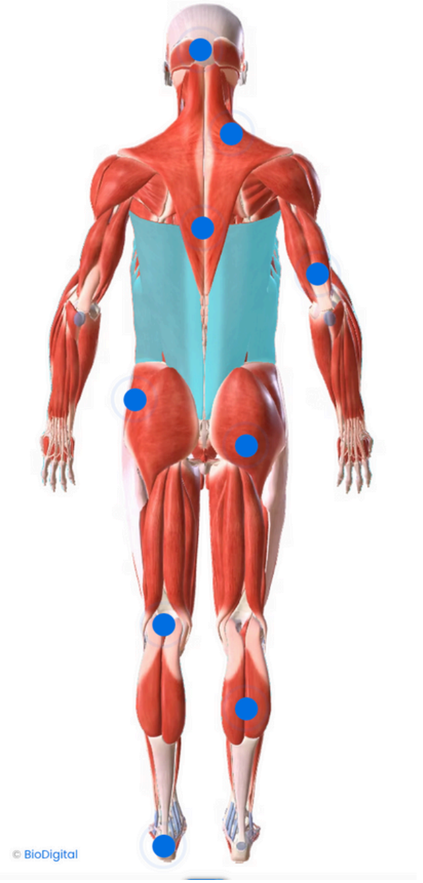


Figure: Pain anamnesis example from Eversion

Methodology: Pain Anamnesis Classification (Binary) - Dense NN

Model Architecture: A deep feed-forward neural network with **7 hidden layers (fully-connected)**. Each hidden layer uses ReLU activations; final layer outputs a vector of probabilities for each pain region's binary label.

The model is essentially performing multi-label classification – predicting multiple binary outcomes simultaneously (e.g. pain in left forefoot, right forefoot, left elbow, etc.)

Loss function: Binary Cross-Entropy with Logits (BCEWithLogitsLoss) aggregated over all output nodes.

Training uses Adam optimizer **(lr=0.001)** with learning rate scheduling (ReduceLROnPlateau) for stability, and gradient clipping to avoid exploding gradients.

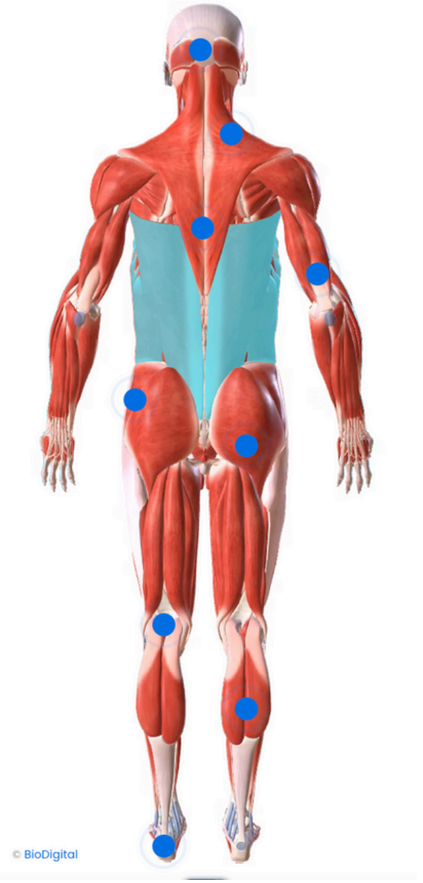


Figure: Pain anamnesis example from Eversion

Results: Pain Anamnesis Classification (Binary) - Dense NN

Binary-Only Model: The deep neural network that predicted pain presence (yes/no) for each region achieved an **overall accuracy ~78%** in its multi-label predictions.

This means about 78% of all individual pain indicators were correctly classified. Given the baseline complexity **(18 outputs simultaneously)**, this accuracy indicates the model learned meaningful patterns linking the input biomechanical measures to pain outcomes.

However, 78% also leaves room for improvement – some pain locations were likely predicted less reliably, possibly those with fewer positive cases in the data (class imbalance could play a role here as well)

Methodology: Pain Anamnesis Classification (Binary + Ordinal) - Dense NN

Motivation: To leverage the full information in pain scores, a second approach was developed using multi-task learning to predict both pain presence (binary) and pain severity (ordinal 0–5) for each relevant body region.

Multi-task learning improves generalization by sharing representations between related tasks [1] – here, the presence and intensity of pain are related tasks.

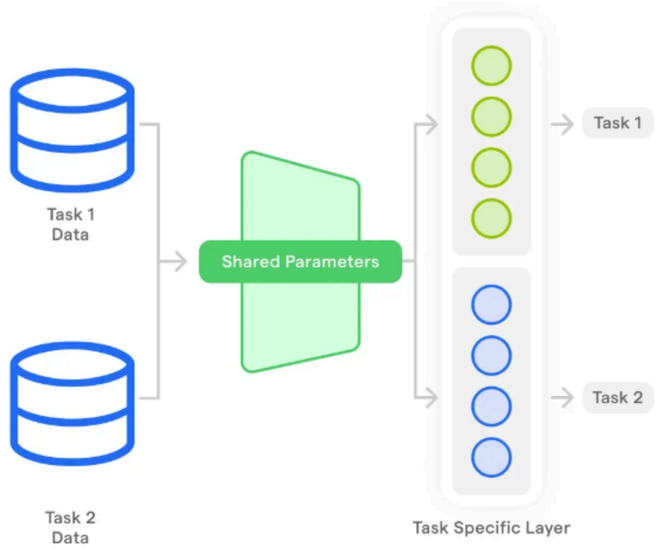


Figure: Multi-task learning model illustration

[1] Multi-task learning

Methodology: Pain Anamnesis Classification (Binary + Ordinal) - Dense NN

Architecture (Shared Trunk NN): A custom MultiTaskModel with a shared feature extractor and separate task-specific output heads.

- Shared Layers: Three fully connected layers (sizes 7 → 64 → 32 → 32) with LeakyReLU activations, batch normalization, and dropout.
- These layers learn a common embedding of the input features relevant to both tasks.

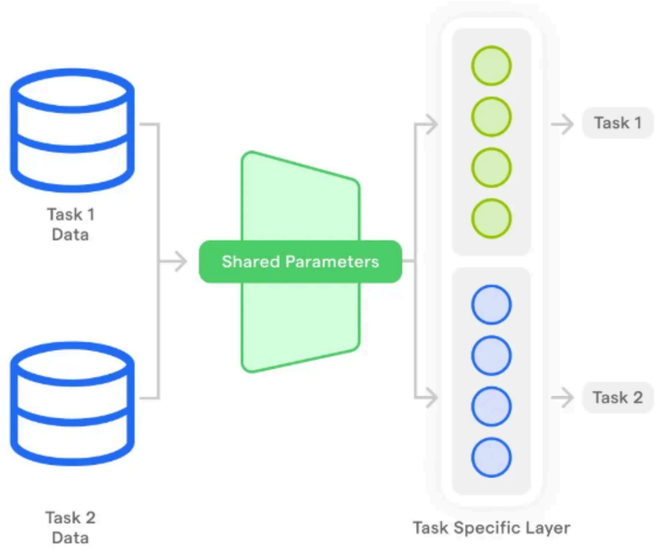


Figure: Multi-task learning model illustration

[1] Multi-task learning

Methodology: Pain Anamnesis Classification (Binary + Ordinal) - Dense NN

Architecture (Shared Trunk NN): A custom MultiTaskModel with a shared feature extractor and separate task-specific output heads.

- Binary Classification Head: A fully connected layer that outputs logits for each binary pain indicator (same 18 outputs as the binary-only model). These logits go through sigmoid to predict pain presence (0/1) for each region.
- Ordinal Regression Heads: For each ordinal pain measure (regions that have a 0–5 rating), a separate sub-network produces an ordinal output. Each ordinal head is implemented via the CORAL (Consistent Rank Logits) approach.
- Instead of directly predicting 0–5, the model outputs (K-1) logits (here (K=6) classes, so 5 logits) per ordinal target, representing the probability that pain is above each severity threshold.
- A given ordinal rating is then determined by counting how many thresholds are passed (i.e., how many logits > 0.5).

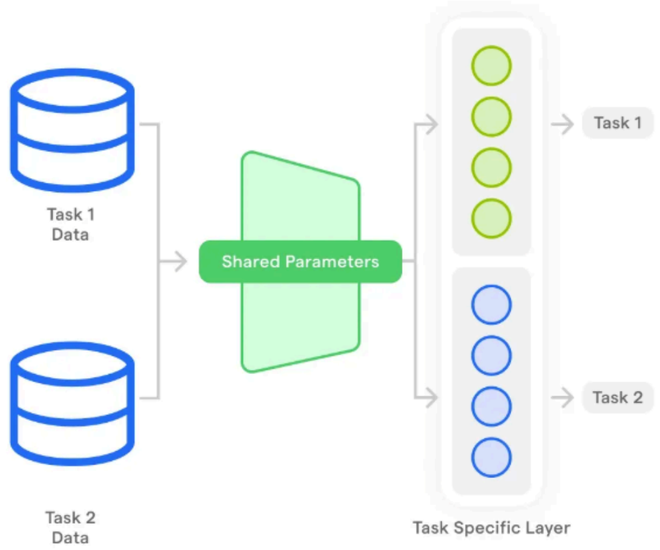


Figure: Multi-task learning model illustration

[1] Multi-task learning

Methodology: Pain Anamnesis Classification (Binary + Ordinal) - Dense NN

Training Details: Both heads are trained jointly. The total loss = binary classification loss + ordinal regression loss (weighted equally). We used binary cross-entropy for binary outputs and a custom CORAL loss (which is also a binary cross-entropy applied to each threshold output).

An epoch's training optimizes both tasks together. Early stopping/validation monitors a combined metric. This multi-task setup allows the model to share learning: e.g. if certain gait features correlate with presence of knee pain, that helps predict intensity too, and vice versa

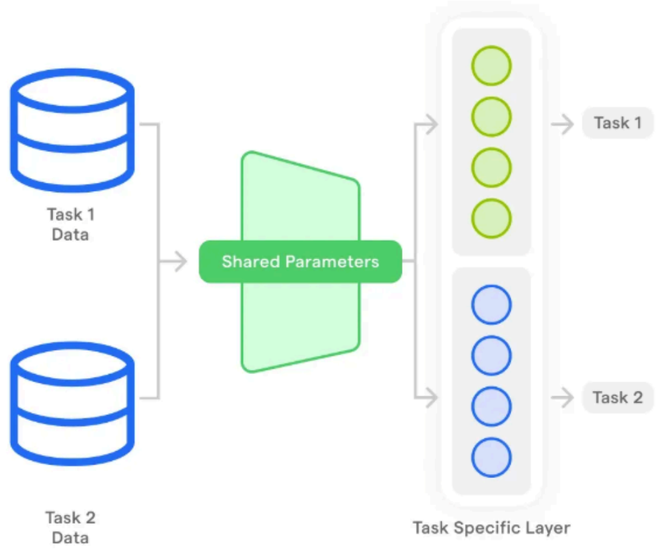


Figure: Multi-task learning model illustration

[1] Cao et al. (2019), Rank consistent ordinal regression for neural networks with application to age estimation

Methodology: Pain Anamnesis Classification (Binary + Ordinal) - Dense NN

Benefit: This approach preserves granular pain information. The binary prediction benefits from the model's awareness of severity, and the ordinal prediction is anchored by learning overall pain occurrence. Multi-task learning with shared layers can improve overall performance by using the inductive bias from related tasks.

In our case, it indeed yielded slightly higher accuracy on pain presence and reasonable error on intensity compared to a binary-only model.

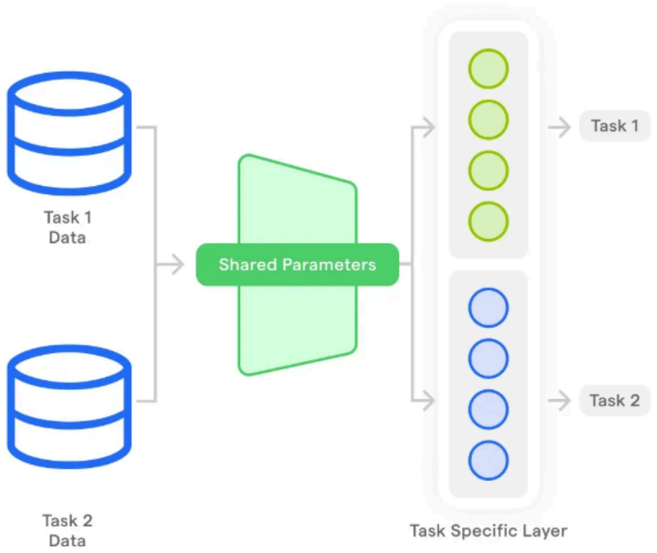


Figure: Multi-task learning model illustration

Results: Pain Anamnesis Classification (Binary + Ordinal) - Dense NN

Combined Binary+Ordinal Model: The multi-task model improved the binary classification performance to **~80% accuracy** on the pain presence task. In addition, it was able to predict pain intensity with a **mean absolute error (MAE) of ~1.4 on a 0–5 scale**.

An MAE of 1.4 implies that, on average, the predicted pain level was within about 1.4 points of the true value – for example, actual “5” (severe) might be predicted as “4”, or actual “2” as “1” or “3” in many cases.

This is a reasonable accuracy for ordinal regression in a medical context, given subjective pain reports can vary.

Results: Pain Anamnesis Classification (Binary + Ordinal) - Dense NN

Benefit of Multi-Task Approach: Despite only a modest increase in binary accuracy **(~2% absolute)**, the combined model provides richer output (it doesn't just say if pain exists, but also estimates its severity). Importantly, it does so in one integrated framework, rather than needing separate models.

The slight accuracy boost suggests that incorporating ordinal data helped the model's overall learning – the network's shared representation likely captured pain patterns better by also accounting for severity distinctions. This reflects the general findings in research that multi-task learning can improve performance by leveraging shared information [1].

Additionally, consistency between tasks was maintained: e.g., the model would rarely predict a high pain intensity while simultaneously predicting "no pain" for that region.

[1] Multi-task learning

Anamnesis Analysis - XGBoost : Data (1)

Data Statistics

- **6 features** related to gait patterns from IMU sensor data (including shoe size)
- **42 target columns**
  - 18 binary pain indicators localized to the distal regions of the lower/upper extremities.
  - 24 non-binary pain indicators localized to multifocal musculoskeletal pains accompanied by cognitive disturbances (e.g., restless sleep, lack of drive, etc).
    - *For the XGBoost model, non-binary columns are binarized (values > 1 : 1, otherwise 0).*

	Misalignment_Movement_Avg_left	Misalignment_Movement_Avg_right	Misalignment_Rest_Avg_left	Misalignment_Rest_Avg_right	Gait_Avg_left	Gait_Avg_right	ShoeSize
0	1.2	0.8	2.1	-0.2	3.2	3.1	NaN
1	0.6	2.2	0.5	3.4	3.3	2.8	37.0
2	0.2	2.0	0.4	2.7	3.7	3.6	37.0
3							
4							
...							
2598							
2599							
2600							
2601							
2602							

2603 rows × 49 columns

feature columns

	Pain_Forefoot_Left	Pain_Forefoot_Right	Pain_Midfoot_Left	Pain_Midfoot_Right	Pain_Heel_Left	...	Pain_Upper_Arm_Right	Pain_Thumb_Left	Pain_Thumb_Right	Pain_Forearm_Left	Pain_Forearm_Right
0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
1	0.0	1.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0									
4	0.0	0.0									
...	...	...									
2598	1.0	1.0									
2599	1.0	1.0									
2600	0.0	0.0									
2601	0.0	0.0									
2602	0.0	0.0									

603 rows × 18 columns

binary target columns

	Pain_Forehead	Pain_Crown	Pain_Neck_Muscles	Pain_Cervical_Spine	Pain_Thoracic_Spine	...	Dizziness_Fall_Tendency	Body_Vibration	Restless_Sleep	Lack_Of_Drive	Concentration_Difficulty
0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
1	0.0	0.0	3.0	3.0	5.0	...	0.0	1.0	3.0	2.0	1.0
2	0.0	0.0	3.0	3.0	5.0	...	0.0	1.0	3.0	2.0	1.0
3	3.0	0.0	3.0	3.0	0.0	...	0.0	0.0	3.0	0.0	3.0
4	0.0	0.0	3.0	3.0	0.0	...	0.0	0.0	4.0	1.0	2.0
...	...	...	...	...	...	...	...	...	...	...	...
2598	1.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2599	1.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2600	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	0.0	0.0
2601	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	2.0	0.0	0.0
2602	0.0	0.0	4.0	3.0	1.0	...	0.0	0.0	2.0	0.0	0.0

2603 rows × 24 columns

non-binary target columns

Anamnesis Analysis - XGBoost : Data (2)

Challenges

- There is **almost no correlation between features and targets.** [1]
  - Pairs of feature values and non-binary / binary target values showed little or no correlations.
    - **Point-Biserial correlation < |0.3|** & **Kendall's Tau < |0.3|**
- There is **little or no correlation within target columns**
  - Most of pairs in non-symmetric relation features showed small **Cramer's V values (< |0.2|).**
- **No linear patterns were observed** between features and target columns.

Model Selection Strategy

- This is a **multi-class and multi-output classification** problem. There are two commonly used approaches to such problems; **multi-task learning(MTL)** and **respective models** for each target column.
- However, due to its nature of internally shared parameters, if tasks are not related or unclear, MTL might lead to negative performance[2][3].
- In such *low correlation scenarios*, where *no linear relationships* are observed, **tree-based models** can perform effectively due to their ability to identify latent hierarchical interactions and stay robust to noise and irrelevant features, unlike target dependent machine learning models .

[1] Measure of Association https://journals.sagepub.com/doi/pdf/10.1177/8756479308317006
[2] Caruana, R. (1997). Multitask learning. Machine learning, 28, 41-75.
[3] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098.

Anamnesis Analysis - XGBoost : Methodology

Model Structure

- **42 respective models for each target column.**
- Each model is trained independently from each other.
  - No shared parameters.

- The model was trained to improve **accuracy**, using **binary cross entropy** as the loss function.

- To compensate for the highly imbalanced binary data, **scale_pos_weight** hyperparameter is set in a broad search space, which adjusts the weight of positive examples in binary classification tasks, which in our case, improved the model's performance more than using resampling techqniques such as SMOTE.

```
params = {
    'early_stopping_rounds': trial.suggest_int('early_stopping_rounds', 50, 100),
    'max_depth': trial.suggest_int('max_depth', 3, 10),
    'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.15),
    'n_estimators': trial.suggest_int('n_estimators', 300, 800),
    'min_child_weight': trial.suggest_int('min_child_weight', 0, 10),
    'subsample': trial.suggest_float('subsample', 0.7, 1.0),
    'colsample_bytree': trial.suggest_float('colsample_bytree', 0.7, 1.0),
    'scale_pos_weight': trial.suggest_int('scale_pos_weight', 1, 10), # Handles the binary class imbalance
    'gamma': trial.suggest_float('gamma', 0, 2),
}
```

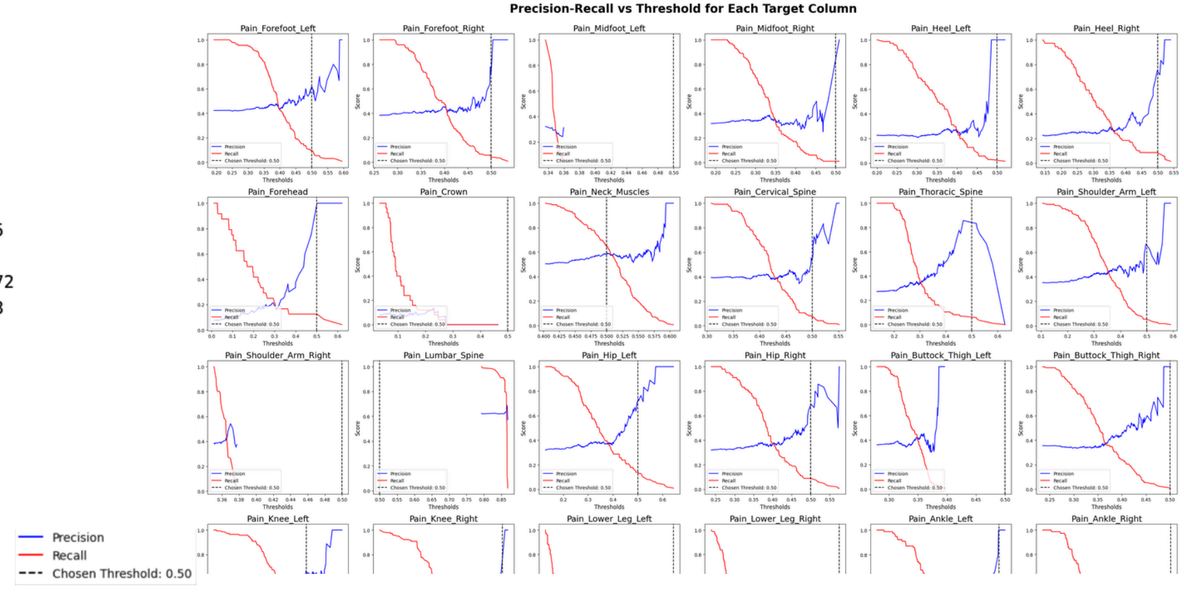
Anamnesis Analysis - XGBoost : Result

Test Result

- The model achieved an overall **accuracy ~74.5%,** but, the model showed a poor **recall of ~9%**.

- The current model used the default threshold of 0.5 to classify the positive classes. However, as shown in the plots below, the recall values start to drop from the beginning, indicating that the current threshold is classifying every instance as negative, increasing the number of false negatives.

```
Mean Test Accuracy: 0.7455
Mean Test Recall: 0.0953
Mean Test Precision: 0.2872
Mean Test F1 Score: 0.1173
Mean Test AUC: 0.5456
Mean Test AUCPR: 0.3064
```



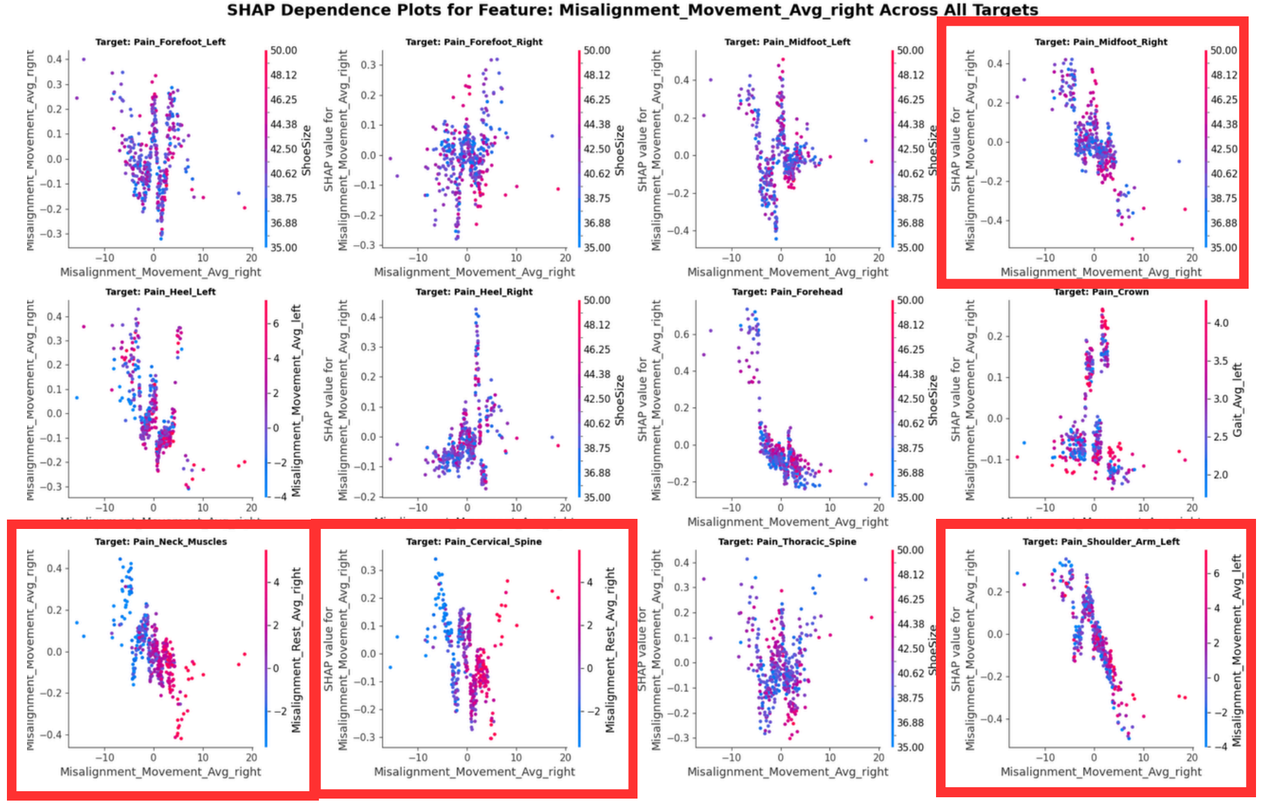
	Target	Accuracy	Recall	Precision	F1_Score	AUC	AUCPR
0	Pain_Forefoot_Left	0.629442	0.197279	0.508772	0.284314	0.578176	0.444836
1	Pain_Forefoot_Right	0.609137	0.244755	0.432099	0.312500	0.587998	0.423628
2	Pain_Midfoot_Left	0.670051	0.074074	0.666667	0.133333	0.586072	0.446938
3	Pain_Midfoot_Right	0.677665	0.109375	0.518519	0.180645	0.566201	0.401328
4	Pain_Heel_Left	0.771574	0.000000	0.000000	0.000000	0.512025	0.269092
5	Pain_Heel_Right	0.776650	0.000000	0.000000	0.000000	0.489973	0.240919
6	Pain_Forehead	0.949239	0.000000	0.000000	0.000000	0.454412	0.074800
7	Pain_Crown	0.936548	0.000000	0.000000	0.000000	0.589756	0.142157
8	Pain_Neck_Muscles	0.527919	0.537234	0.505000	0.520619	0.535685	0.514389
9	Pain_Cervical_Spine	0.583756	0.270440	0.472527	0.344000	0.553780	0.467193
10	Pain_Thoracic_Spine	0.733503	0.085714	0.500000	0.146341	0.575284	0.340814
11	Pain_Shoulder_Arm_Left	0.692893	0.075630	0.450000	0.129496	0.551658	0.381818
12	Pain_Shoulder_Arm_Right	0.659898	0.112903	0.368421	0.172840	0.559170	0.387990
13	Pain_Lumbar_Spine	0.652284	0.937500	0.664820	0.777958	0.510360	0.645214
14	Pain_Hip_Left	0.619289	0.063830	0.333333	0.107143	0.580607	0.408401
15	Pain_Hip_Right	0.649746	0.105263	0.424242	0.168675	0.552358	0.400604
16	Pain_Buttock_Thigh_Left	0.662437	0.031746	0.266667	0.056738	0.529081	0.346609
17	Pain_Buttock_Thigh_Right	0.644670	0.095588	0.433333	0.156627	0.551898	0.379902
18	Pain_Knee_Left	0.568528	0.304878	0.471698	0.370370	0.541914	0.474667
19	Pain_Knee_Right	0.543147	0.238095	0.555556	0.333333	0.550858	0.530724
20	Pain_Lower_Leg_Left	0.756345	0.000000	0.000000	0.000000	0.539884	0.298702
21	Pain_Lower_Leg_Right	0.748731	0.000000	0.000000	0.000000	0.578326	0.314866
22	Pain_Ankle_Left	0.776650	0.073684	1.000000	0.137255	0.573983	0.342972
23	Pain_Ankle_Right	0.761421	0.020833	1.000000	0.040816	0.600496	0.353845
24	Restless_Legs_Syndrome	0.794416	0.000000	0.000000	0.000000	0.480022	0.221458
25	Dizziness_Fall_Tendency	0.847716	0.000000	0.000000	0.000000	0.613922	0.222074
26	Body_Vibration	0.926396	0.000000	0.000000	0.000000	0.494757	0.077807
27	Restless_Sleep	0.553299	0.359788	0.552846	0.435897	0.609808	0.586166
28	Lack_Of_Drive	0.723350	0.009524	0.166667	0.018018	0.428538	0.233616
29	Concentration_Difficulty	0.741117	0.000000	0.000000	0.000000	0.503760	0.264720
30	Pain_Wrist_Left	0.860406	0.000000	0.000000	0.000000	0.594905	0.179565
31	Pain_Wrist_Right	0.791878	0.012195	0.500000	0.023810	0.553979	0.249548
32	Pain_Elbow_Left	0.903553	0.000000	0.000000	0.000000	0.551375	0.110473
33	Pain_Elbow_Right	0.873096	0.000000	0.000000	0.000000	0.509186	0.136723
34	Pain_Finger_Left	0.766497	0.010753	1.000000	0.021277	0.520344	0.268184
35	Pain_Finger_Right	0.746193	0.031579	0.272727	0.056604	0.521246	0.265766
36	Pain_Upper_Arm_Left	0.860406	0.000000	0.000000	0.000000	0.527755	0.166006
37	Pain_Upper_Arm_Right	0.829949	0.000000	0.000000	0.000000	0.531654	0.193445
38	Pain_Thumb_Left	0.855330	0.000000	0.000000	0.000000	0.585194	0.223292
39	Pain_Thumb_Right	0.814721	0.000000	0.000000	0.000000	0.594078	0.240565
40	Pain_Forearm_Left	0.918782	0.000000	0.000000	0.000000	0.532631	0.086683
41	Pain_Forearm_Right	0.903553	0.000000	0.000000	0.000000	0.510571	0.111260

Anamnesis Analysis - XGBoost : Model interpretation

Despite extensive data analysis, no explicit relationships between the features and the targets were identified, and the model's performance did not improve.

Instead building the model from the data, understanding how the features affected the model’s prediction values can help improve the model’s performance.

SHAP allows us to examine Shapley values. In particular, analyzing the SHAP dependence plots can help us identify which features have the greatest impact on the model’s predictions.



Conclusion

Summary of Achievements: We demonstrated that machine learning models can significantly enhance two disparate yet related biomechanical analysis tasks.

- An ML-based approach accurately detects gait events from wearable sensor data, outperforming legacy threshold-based algorithms and operating robustly across conditions.

- Likewise, an ML model can automatically classify patient pain outcomes (across multiple body regions) and even estimate pain severity, which traditionally required manual assessment.

- Furthermore, post training analysis using interpretable machine learning tools give a room to further diagnose a trained model and effectively improve the performance.

Future Directions

Gait Model: Address the rare-event problem (heel strike) by data augmentation or a hybrid model that ensures those events are recognized. Additionally, exploring a CNN-LSTM hybrid or Temporal Convolution Network could further boost accuracy and timing precision.

Pain Model: Refine the ordinal prediction, possibly using a larger dataset or incorporating demographic factors. Also, evaluate the model on new patient data to ensure generalizability. An interesting extension is to predict pain trajectory (e.g. risk of pain increasing or decreasing) using sequential models on follow-up anamnesis data.

Integration: Ultimately, combine gait and pain analysis – e.g., use gait features not just from static measures but from dynamic gait signals (like how the person walks) to improve pain predictions. This could lead to a holistic system that links functional gait patterns with self-reported pain, giving a fuller picture of a patient's condition.

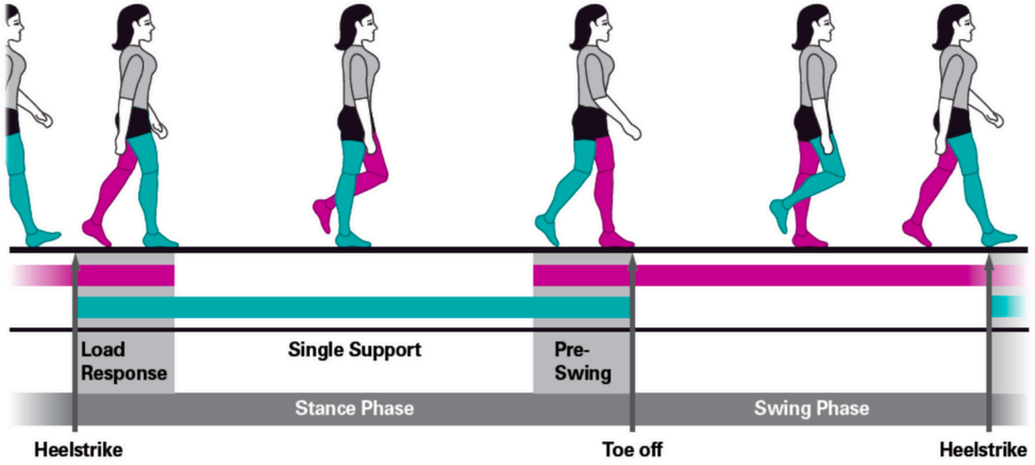
Conclusion

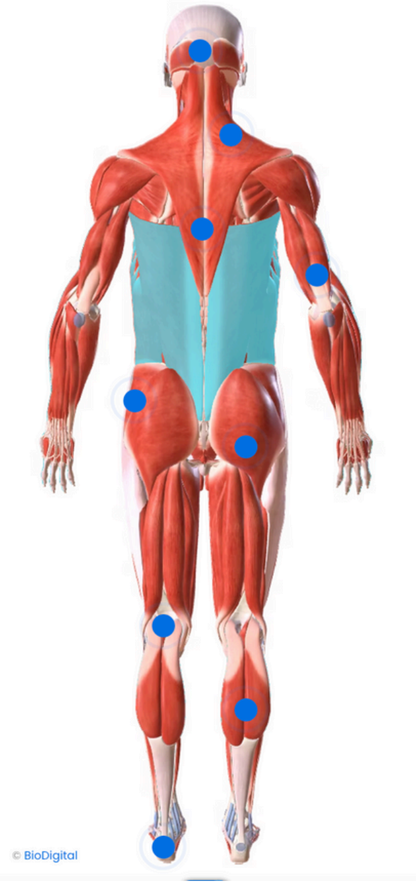
Impact: These ML approaches offer greater automation and objectivity.

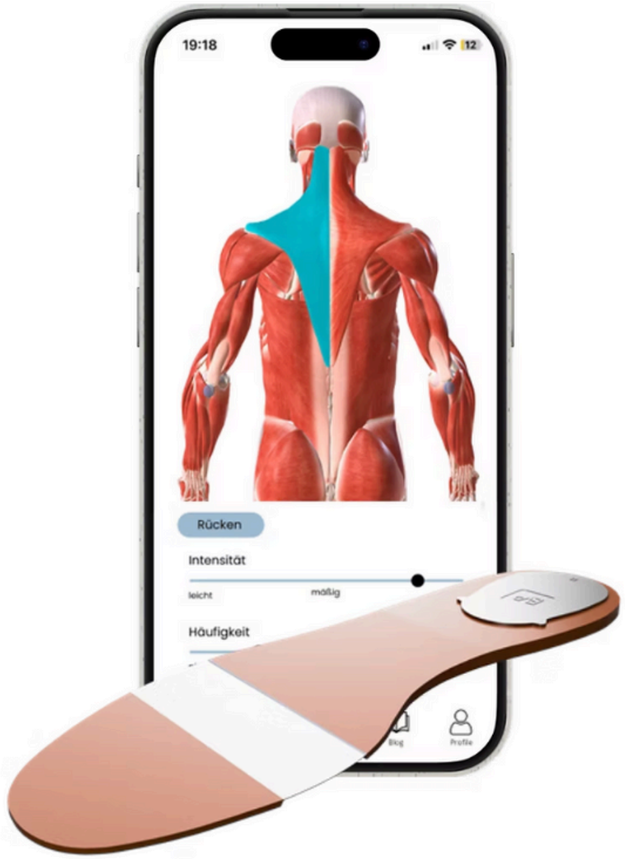
- For gait analysis, this means clinicians or engineers can get real-time phase detection without setting up force plates – useful in rehab settings or prosthetic tuning on the go.

- For pain anamnesis, it means leveraging routinely collected measurements (e.g. foot pressure, alignment data) to predict pain based on gait analysis, potentially aiding early detection of issues or personalized therapy recommendations.

Thank you!


Load Response
Single Support
Pre-Swing
Stance Phase
Swing Phase
Heelstrike
Toe off
Heelstrike


© BioDigital


19:18
Rücken
Intensität
leicht
mäßig
Häufigkeit

Expected questions:

- What are the current methods used at Eversion for gait event classification and pain anamnesis?
  - EVERSION currently uses static algorithms based on graph analysis (e.g., detecting peaks, saddle points, and inflection points) for gait event detection. However, there is no existing approach for correlating pain anamnesis with gait parameters.
- What are the challenges with the current methods?
  - The current gait event detection methods can be sensitive to individual gait variations and challenging edge cases, such as running or atypical walking patterns. Additionally, pain anamnesis relies solely on self-reported data, which is inherently subjective and may lack consistency across different users.
- How does this project help with these challenges?
  - By implementing machine learning models such as LSTM and XGBoost, this project aims to enhance the accuracy and adaptability of gait event detection. Moreover, integrating objective gait data with pain reports allows for a more comprehensive and reliable assessment of musculoskeletal health, reducing subjectivity in pain analysis. Additionally, the ML approach enables cross-checking the expected pain areas—predicted from gait parameters—with the actual pain areas reported by the user. This validation step can help identify inconsistencies, refine pain assessments, and improve the overall reliability of pain anamnesis.
- How does this project facilitate transitioning from current methods to ML-based methods?
  - For EVERSION the results of this project are a good basis to improve their system performance and reliability by switching to ML approaches in future.
- What will be the future directions for Eversion based on the results of this project?
  - EVERSION plans to further refine the ML models using additional data, including validation measurements, customer-reported pain anamnesis, and gait parameter data. The ultimate goal is to replace the current static gait event detection algorithms with more robust, faster, and more reliable ML-based methods
- Does this project increase confidence in ML-based techniques to accomplish these tasks?
  - Yes, the results—especially for gait event detection—strongly validate the potential of ML-based approaches. The project has confirmed that machine learning can provide significant improvements in accuracy, adaptability, and robustness, reinforcing EVERSION’s confidence in transitioning to ML-driven solutions.

Expected questions:

- Are there other methods being tried at Eversion or in the industry that show comparable/promising results?
  - ??
- Are there cost benefits to using ML-based methods?
  - Implementing ML-based methods can reduce the need for extensive manual assessments and improve early detection of musculoskeletal issues, potentially lowering long-term healthcare costs.
- How do ML-based methods handle variability in individual gait patterns?
  - ML models like LSTM can learn temporal dependencies and adapt to individual gait variations, improving detection accuracy over traditional methods.
- What data privacy measures are in place when collecting and analyzing patient data?
  - Ensuring compliance with data protection regulations, such as GDPR, is crucial when handling sensitive health information in ML applications.
- How do you validate the accuracy of the ML models used in this project?
  - Model validation involves using labeled datasets to assess performance metrics like accuracy, precision, recall, and conducting cross-validation to ensure robustness.
- Can these ML models be integrated into wearable devices for real-time monitoring?
  - Yes, with optimization, ML models can be embedded into wearable technologies to provide continuous, real-time gait and pain monitoring.
- What are the limitations of using ML in gait and pain analysis?
  - Challenges include the need for large, diverse datasets for training, potential model overfitting, and ensuring interpretability of the ML models' decisions.