Feasibility of Augmenting Gait Analysis with Machine Learning Models

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Technology

IN

Instrumentation System

BY

Tejas Doypare

Under the Guidance of

Prof. Shivakumar Sastry

Instrumentation and Applied Physics
Indian Institute of Science
Bangalore - 560 012 (INDIA)

2025-05-26

Declaration of Originality

I, **Tejas Ramdas Doypare**, with SR No. **22874** hereby declare that the material presented in the thesis titled

Feasibility of Augmenting Gait Analysis with Machine Learning Models

represents original work carried out by me in the **Instrumentation and Applied Physics** at **Indian Institute of Science** during the years **2024-25**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Advisor Signature

Abstract

This thesis presents the results for exploration of a machine learning-based system for human gait analysis using low-cost, vision-only components, RGB cameras, and MediaPipe pose estimation. The system captures spatiotemporal features from pose landmarks and predicts Center-of-Pressure (CoP) trajectories using a combination of classical models (Linear Regression, MLP, Random Forest, AdaBoost), deep sequential architectures (LSTM and Seq2Seq) and Retrieval Augmented Generation (RAG).

Key contributions include implementing spatial registration using SVD-based affine transformation between image space and real-world coordinates, data augmentation through interpolation of static footstep images, and comprehensive evaluation of models on CoP regression. Experimental results show that temporal models, particularly Seq2Seq and LSTM variants, achieve sub-centimeter mean squared error relative to GAITRite ground truth, outperforming all static baselines. This work identified a few critical gaps and lays the foundation for low-cost, accessible gait diagnostics using vision-based machine learning.

Keywords: Gait Analysis, Center of Pressure (CoP), Pose Estimation, Machine Learning, LSTM, RAG, Spatial Registration

Acknowledgements

I would like to express my heartfelt gratitude to **Prof. Shivakumar Sastry** for his continuous support, guidance, and encouragement throughout this project. His deep knowledge, valuable feedback, and constant support played a key role in shaping my understanding and improving the quality of my work. I truly appreciate the time and effort he invested in helping me overcome challenges and stay on the right track.

I am sincerely thankful to **Prof. Tapajyoti Das Gupta** for his honest and constructive suggestions, which helped me identify key areas for improvement and guided me in the right direction throughout the project.

A special thanks to **Dr. Albert** and **Banashree ma'am** for their kind support in facilitating the collection of actual gait data using the GAITRite system at the Centre for Brain Research (CBR), IISc Bangalore. Their help was instrumental in the successful execution of the experimental phase of this work.

I am also deeply grateful to **Vikas**, **Ashish**, **Suprit**, **Sudhanshu**, **Ayush**, **Swapnil**, and all my friends for their constant support and encouragement. Your presence, advice, and help at every step made this journey meaningful and memorable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Human gait is the manner or style of walking. It is a rich source of information about a person's neuromuscular health. Quantitative gait analysis is used clinically to assess conditions such as Parkinson's disease, post-stroke recovery, cerebral palsy, and age-related fall risks. A critical parameter in such assessments is the **Center of Pressure (CoP)** [16], which captures the point of application of the resultant ground reaction force [5]. This is a sequence of coordinate positions from which the Gaitrite system derives all the critical Gait Parameters.

Traditional gait analysis systems such as **GAITRite** offer sub-centimeter CoP resolution using pressure-sensitive walkways. However, they are expensive, non-portable, and require specialized installation. In contrast, consumer-grade RGB cameras and pose estimation tools like **MediaPipe** [13] and low-cost IMU sensors have democratized access to high-quality 3D human joint position data [23]. This convergence raises an important research question:

> *Can we accurately predict gait parameters using only affordable, non-intrusive sensors and machine learning?*

This thesis, however, used only low-cost cameras because of the non-availability of data from IMU sensors in a timely manner.

1.2 Research Objectives

The primary goal of this thesis is to explore a vision-based approach, augmented with machine learning system that replicates GAITRite's Center of Pressure trajectories using pose data extracted from static images captured by RGB cameras.

Specific objectives:

1. Capture gait sequences from static images using a dual-camera setup and calibrate them to real-world coordinates.

2. Extract 3D joint landmarks from image frames using MediaPipe Pose.

3. Register pose landmarks with GAITRite ground-truth CoP data using affine transformation.
4. Generate synthetic data via interpolation to compensate for missing sequences.
5. Evaluate a suite of classical and deep learning models for CoP prediction, including:
    - Static regressors: Linear Regression, MLP, Random Forest, AdaBoost
    - Sequential models: LSTM, and Seq2Seq (Encoder-Decoder)
    - RAG model
6. Quantify prediction accuracy via MSE and visualize the predicted CoP paths against ground truth.

1.3 Scope of the Thesis

1. **Input:** Extracted Vision-based 3D pose landmarks from MediaPipe from static walking images.
2. **Output:** Sequence of CoP (X, Y) coordinates for each footstep.
3. **Hardware:** Two RGB cameras, calibration poster, and GAITRite system for ground truth.
4. **Focus:** Model training, registration pipeline, interpolation, sequence modeling.
5. **Exclusion:** No real-time deployment.

1.4 Challenges Encountered

1. **Synchronization issues:** Two dual cameras were not synchronized. For effective Gait analysis, visual feed from two synchronized cameras at high data frame rates is needed. Since this equipment is not available, we used static images with interpolated data to explore the analysis and processing techniques.
2. **Sparse dataset:** Gaitrite systems are very expensive and can only provide limited data for footstep. In this work only 7 usable footsteps were collected. This was augmented with synthetic data.
3. **Pose noise:** MediaPipe outputs may have position noise especially for foot joints.
4. **Registration complexity:** Required accurate spatial transformation between image space and physical CoP space.

1.5 Contributions

1. Designed a complete vision-based gait data acquisition pipeline using affordable cameras.
2. Implemented a spatial registration algorithm using the SVD-based affine transformation between virtual and real-world frames [19].
3. Created an interpolated dataset from discrete static frames to simulate continuous gait sequences.
4. Trained and compared a wide range of machine learning models, both static and temporal, on the CoP prediction task.
5. Demonstrated that the LSTM-based [8] and Seq2Seq models [22] can be an approach to GAITRite-level accuracy.

This thesis aims to bridge the gap between clinical grade gait systems and accessible ML-based tools, allowing greater deployment of gait diagnostics in healthcare and rehabilitation.

Chapter 2

Theoretical Background

This chapter introduces the fundamental biomechanical and machine learning concepts underlying this thesis. It includes an overview of human gait phases and spatiotemporal parameters [5], the biomechanical meaning of the Center of Pressure (CoP) [16], and the theoretical motivations for using machine learning models, especially sequential architectures, to model gait dynamics.

2.1 Human Gait Fundamentals

Gait Cycle

The gait cycle represents the complete sequence of movements during walking, starting with the heel-strike of one foot and ending with the subsequent heel-strike of the same foot. It is divided into:

- **Stance Phase (60% of cycle):** The foot remains in contact with the ground and supports the body's weight. This includes heel-strike, mid-stance, and toe-off.
- **Swing Phase (40% of cycle):** The foot moves forward through the air to prepare for the next step.

The precise timing and symmetry of these phases are critical indicators of gait health and pathology.

Spatiotemporal Parameters

These are key biomechanical quantities measurable during walking:

- **Step length:** Distance between the heel strikes of the opposite feet.
- **Stride length:** Distance between consecutive heel strikes of the same foot.
- **Cadence:** Number of steps taken per minute.
- **Velocity:** Walking speed, computed as stride length multiplied by cadence.
- **Step width:** Side-to-side distance between the feet.

Changes in these parameters can indicate neuromuscular disorders, musculoskeletal injury, or aging-related degradation.

2.2 Center of Pressure (CoP)

Concept: The CoP is the point of application of the ground reaction force vector beneath the foot. It reflects balance and pressure distribution during stance.

Clinical Relevance

- A shifting or asymmetrical CoP path may indicate balance disorders, fall risk, or motor impairment.
- In Parkinsonian gait, CoP exhibits reduced progression and increased medial-lateral oscillation.

Measurement Techniques

- **GAITRite:** Commercial walkway system with embedded pressure sensors.
- **Force Plates:** Accurate but non-portable lab-grade devices.
- **Vision-Based Estimation:** Using machine learning on pose-derived features.

2.3 Pose Estimation Theory

Skeleton tracking algorithm or Pose estimation has been very well developed, referring to the extraction of joint coordinates from visual data. In this thesis, we used open source software called Mediapipe to estimate joint positions.

- **2D Pose:** Estimation of (x, y) coordinates in image space.
- **3D Pose:** Includes relative depth (z), enabling spatial reasoning.

MediaPipe Pose returns 33 landmarks per frame using a CNN-based detector and a landmark regression model, which estimates joints in real time from RGB images.

2.4 Machine Learning for Gait Modeling

The theoretical foundations and concepts discussed in this work are adapted from standard textbooks, with full references provided in the References section [2, 20].

Why Machine Learning?

- Gait dynamics are highly non-linear and subject-specific.
- Rule-based systems fail to generalize across populations.
- ML learns complex patterns from pose data to predict clinically relevant outputs such as CoP.

Formal Mapping

Given joint sequences $X = [x_1, x_2, ..., x_T]$ with $x_t \in \mathbb{R}^{28}$, the goal is to learn a function f such that:

$$Y = f(X), \quad Y =< y_1, ..., y_{T'} >, \quad y_t \in \mathbb{R}^2.$$

Static vs. Temporal Models

Static: Predict y_t from single x_t. Ignores context. (e.g., Linear Regression, MLP)

Temporal: Learn from full sequences $(x_1, ..., x_T)$. Capture dynamics. (e.g. LSTM, and Seq2Seq)

2.5 Time-Series Learning Theory

Recurrent Neural Networks (RNN) [17]

Process sequential inputs with memory via hidden state h_t:

$$\begin{aligned} h_t &= \tanh(Wx_t + Uh_{t-1} + b), \\ \hat{y}_t &= Vh_t + c. \end{aligned}$$

Where:

- x_t is the input vector at time t (e.g., pose features).
- h_t is the hidden state.
- W, U are weight matrices for input and recurrent connections, respectively.
- b, c are bias vectors.
- V maps hidden state to output prediction $\hat{y}_t$.

Long Short-Term Memory (LSTM)

Introduces cell state c_t with gates to control information flow:

$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), && \text{(input gate)} \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), && \text{(forget gate)} \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), && \text{(output gate)} \\ \tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c), && \text{(candidate memory)} \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, && \text{(new memory)} \\ h_t &= o_t \odot \tanh(c_t). && \text{(new hidden state)} \end{aligned}$$

Where:

- σ is the sigmoid activation.
- $\odot$ denotes element-wise multiplication.

- i_t, f_t, o_t are gating vectors in $[0, 1]$.
- $\tilde{c}_t$ is the candidate cell content.

LSTM mitigates vanishing gradient by directly connecting past memory cells c_{t-1} to current state c_t.

GRU (Gated Recurrent Unit) [4]

Simpler than LSTM with two gates:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1}), && \text{(update gate)} \\ r_t &= \sigma(W_r x_t + U_r h_{t-1}), && \text{(reset gate)} \\ \tilde{h}_t &= \tanh(W x_t + U(r_t \odot h_{t-1})), && \text{(candidate activation)} \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. && \text{(final output)} \end{aligned}$$

Sequence-to-Sequence (Seq2Seq)

Encodes input sequence to a context vector and decodes future output:

$$\text{Encoder: } h_T = \text{LSTM}(x_1, ..., x_T), \quad \text{Decoder: } \hat{y}_t = \text{LSTM}(\hat{y}_{t-1}, h_t)$$

Often trained with **teacher forcing**, i.e., using ground truth y_{t-1} instead of predicted $\hat{y}_{t-1}$.

2.6 Affine Transformations and Registration

Pose landmarks from MediaPipe are in pixel space (virtual coordinates). CoP ground truth from GAITRite is in physical space (cm). So we required to convert virtual coordinates into the physical coordinates.

It is done by below equation:

$$P_{\text{phy}} = R \cdot P_{\text{vir}} + T.$$

Here:

- P_{phy} is the physical coordinates.
- P_{vir} is the virtual coordinates.
- R is a rotation matrix.
- T is a translation vector.

Firstly, we are fixing the stationary points ie poster points. then we measure physical coordinates of poster points from the one fixed point(any stationary point from the 3D space). then we capture image of those poster from the stable camera location and gets virtual coordinates of the same poster. After getting physical and virtual coordinates of the stable points ie. poster points we do SVD to computes R and T .

2.7 Summary

This chapter has outlined the biomechanical foundations of gait, the mathematical theory of pose estimation and time-series learning, and the need for affine registration. These concepts form the theoretical basis for the gait analysis system presented in this thesis.

Chapter 3

Literature Review

This chapter surveys existing literature across four domains relevant to this thesis: (1) vision-based gait analysis, (2) IMU-based gait analysis, (3) multimodal sensor fusion, and (4) machine learning techniques[11] applied to gait modeling. We also highlight research gaps that motivate our approach.

3.1 Vision-Based Gait Analysis

Early vision-based systems [15] relied on marker-based motion capture (e.g., Vicon) to estimate joint trajectories, but these setups are expensive and constrained to labs. Recent advancements in 2D/3D pose estimation, particularly using deep learning, have enabled markerless systems using standard RGB cameras.

3.1.1 Pose Estimation Frameworks

- **MediaPipe Pose** (Google) provides real-time 3D landmark detection of 33 joints using a lightweight ML model. It has been used in sports, rehabilitation, and animation.
- **OpenPose** (Carnegie Mellon) detects 2D keypoints via Part Affinity Fields. though highly accurate, it is computationally expensive.
- **AlphaPose** and **HRNet** offer state-of-the-art 2D pose accuracy but require GPU-based inference.

3.1.2 Gait Parameter Estimation

Stenum et al. (2021) used 2D ankle trajectories from OpenPose to estimate stride time and stride length with ¡3% error compared to GAITRite.

Jamsrandorj et al. (2022) proposed a CNN–Transformer hybrid for spatio-temporal parameter estimation using single-camera frontal walking videos. Their model achieved $r > 0.93$ correlation for step length and velocity.

3.1.3 Limitations

- Occlusion and viewpoint sensitivity reduce landmark accuracy.

- Depth ambiguity in monocular setups leads to errors in step height or velocity.
- Existing models assume fixed walking directions or constrained backgrounds.

3.2 IMU-Based Gait Analysis

Wearable inertial measurement units (IMUs), containing accelerometers and gyroscopes, enable gait estimation in uncontrolled environments.

Mariani et al. (2013) used shoe-mounted IMUs to track stride length and clearance. Their method achieved a RMS error less than 2cm over 10m walking.

Dehzangi et al. (2017) transformed IMU signals into spectrograms and trained CNNs for subject identification, achieving 96% classification accuracy on 10 subjects.

3.2.1 Advantages

- IMUs are affordable, portable, and independent of lighting.
- Allow real-world gait capture over long durations.

3.2.2 Challenges

- Require calibration and drift compensation.
- Sensitivity to placement and soft tissue motion.

3.3 Multimodal Sensor Fusion

Several studies combine vision and inertial data to improve robustness.

Bijalwan et al. (2021) fused Kinect skeletons with IMU accelerometer data for joint angle estimation. Their hybrid system outperformed vision-only baselines in clinical populations.

Alharthi et al. (2019) combined foot-mounted IMUs with depth cameras and used deep CNNs to classify gait phases. Their fusion model improved F1-score by 8% over unimodal inputs.

3.3.1 Fusion Techniques

- Early Fusion: Concatenate raw sensor streams.
- Late Fusion: Combine the outputs of independent models.
- Cross-modal Attention: Learn attention weights over modalities (future work).

3.4 Machine Learning for Gait Analysis

3.4.1 Classical Approaches

Linear regression and decision trees have long been used to predict gait parameters from hand-crafted features such as stride frequency, ground reaction force, etc.

Random Forests and Boosted Trees improve over linear models by capturing non-linear relationships but lack temporal memory.

3.4.2 Neural Networks and Sequence Models

Multilayer Perceptrons (MLP) can regress directly from pose vectors to CoP but ignore motion continuity.

Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and time-series dynamics across gait cycles.

3.4.3 Seq2Seq for Trajectory Prediction

Sequence-to-sequence architectures, originally designed for language translation, have been adapted for predicting future trajectories. They are particularly useful for forecasting multistep CoP paths conditioned on prior footstep sequences.

3.4.4 Recent Benchmarks

Khandelwal et al. (2022) evaluated several sequence models for predicting joint trajectories from pose sequences and showed that Transformer-based models outperformed LSTM when sufficient data was available.

3.5 Research Gaps

- Most vision-only systems lack proper calibration to map pose data into real-world units.
- Few works use only RGB cameras (without depth) to predict clinical-grade CoP paths.
- Dataset availability remains limited, especially for multimodal sequences with CoP ground truth.
- No prior study combines spatial registration, interpolation, and Seq2Seq prediction in the same pipeline.

3.6 Our Contribution in Context

This thesis addresses the above gaps by:

- Using MediaPipe to extract 3D pose landmarks from dual-camera footage.
- Applying affine registration to align pixel-space to GAITRite coordinates.
- Constructing a synthetic dataset through interpolation of static footstep sequences.
- Training classical and deep sequence models (LSTM, Seq2Seq) to predict CoP with sub-centimeter accuracy.

The following chapters describe the system architecture and modeling pipeline in detail.

Chapter 4

System Architecture and Data Acquisition

This chapter presents the design of the integrated hardware-software pipeline used to collect human gait data. It includes the recording environment, camera configuration, pose extraction, synchronization issues, and the strategy used to link image-based pose landmarks to physical Center-of-Pressure (CoP) values acquired using the GAITRite system.

4.1 System Overview

The goal was to design a low-cost system capable of capturing gait kinematics and synchronizing it with clinically valid CoP trajectories. The key components included:

- **Two RGB cameras and tripod**: Positioned to monitor the walkway from two orthogonal viewpoints.
- **GAITRite Walkway System (Figure 4.1 and 4.2)**: A clinically validated pressure-sensitive mat providing sub-centimeter CoP measurements.
- **MediaPipe Pose**: A vision-based pose estimation framework for extracting 3D joint coordinates.
- **Python-based processing pipeline**: Performs registration, data interpolation, transformation to physical units, and prepares ML-ready datasets.

Figure 4.1: View 1 of the GAITRite system setup, showing the pressure-sensitive walkway and the placement of calibration poster B used for spatial registration.

Figure 4.2: View 2 of the GAITRite system setup, showing the pressure-sensitive walkway and the placement of calibration poster A used for spatial registration.

Figures 4.1 and Figure 4.2 show images of the actual GAITRite system located at the Centre for Brain Research (CBR), IISc Bangalore. As seen in the images, two calibration posters were set up to facilitate camera calibration for the gait capture system.

4.2 Camera Setup

Two mobile phone cameras were used for passive capture of the subject's walking motion. The key characteristics of the setup were:

- The cameras were mounted at approximately 1.2 meters height and tilted downward at a 30-degree angle.
- Each camera had a fixed field of view spanning a 2.5 meter section of the walkway.
- The frames were captured as individual static images due to the lack of synchronization hardware.
- Calibration was performed using a printed dot poster (Figure 4.3) with known real-world coordinates. (red points are actual points detected by MediaPipe, blue are predicetd points)

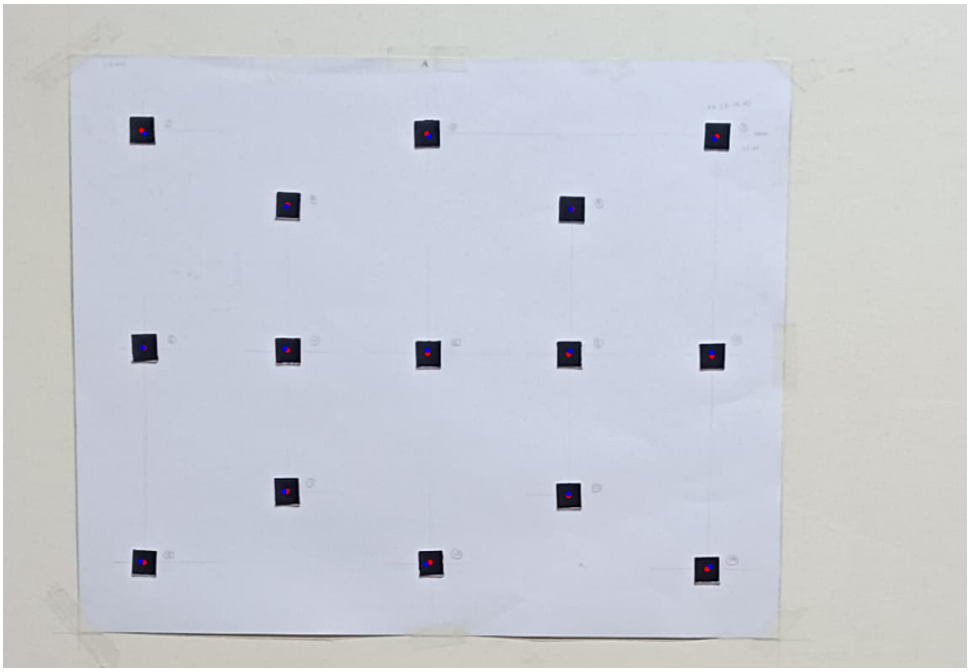

Figure 4.3: Calibration poster with a fixed grid of 15 printed dot markers used for spatial registration between camera image coordinates and real-world physical coordinates.

Technical Specifications

- **Resolution:** 4080x3060
- **Frame Rate:** Captured at approximately 1 fps, because we collected static images data and not video due to the synchronization problem.
- **Acquisition Method:** Captured and saved in my local system.

4.3 GAITRite System

The GAITRite system (refer Figures 4.1 and 4.2) consist of an electronic mat with pressure sensors arranged in a grid pattern. As the subject walks across the mat:

- The mat logs precise foot contacts and pressure distributions.
- CoP (X, Y) is calculated based on the weighted average of pressure sensors activated during stance.
- Temporal gait parameters (step duration, and cadence) are also recorded.

These CoP values serve as the ground truth for model training and are linked to pose sequences extracted from images.

4.4 Pose Estimation via MediaPipe

MediaPipe Pose extracts 3D coordinates for 33 anatomical landmarks (fig.4.3), including: `nose`, `shoulders`, `hips`, `knees`, `ankles`, `heels`, and `toes`.

Each output frame provides the following:

- x, y, z coordinates in normalized image space (i.e., [0, 1] range).
- A confidence score for visibility.
- Relative depth values (z) to capture the forward motion of the subject.

To transform these normalized outputs into physical units, a spatial calibration (described in the next section) was applied.

4.5 Calibration and Spatial Registration

Pose landmarks from MediaPipe are output in image-normalized coordinates. To relate these to CoP values recorded by GAITRite (in cm), an affine transformation was computed:

- A calibration board with known physical dot coordinates was captured by the camera.
- Image-space coordinates Q of these dots were extracted using OpenCV.
- World-space coordinates P were already known.
- Transformation $P_{\text{real}} = R \cdot P_{\text{pixel}} + T$ was computed using Singular Value Decomposition (SVD).

This transformation was applied to all MediaPipe landmarks to project them into the physical CoP space.

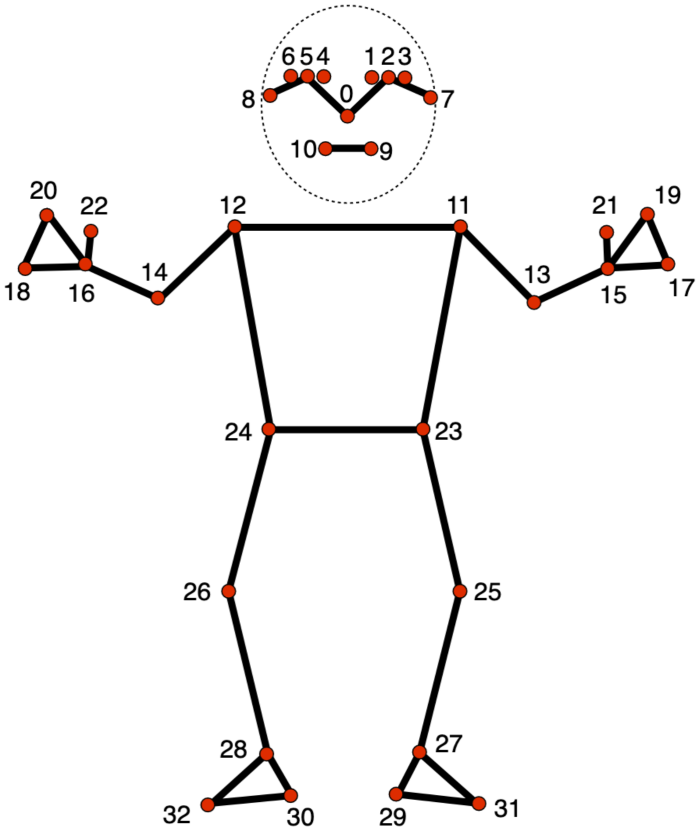

Figure 4.4: Detected landmark points on the human body extracted using MediaPipe, representing key joints such as shoulders, wrists, hips, and heels for gait analysis.

4.6 Synchronization and Data Alignment

Synchronization Challenge

If we capture the video of person walking on the GAITRite system. then its hard to synchronize the capture videos from two different positions and different cameras.

Workaround:

- Instead of recording continuous walking, static image snapshots were taken for each individual step.

Limitation: This reduced the temporal resolution of the data and required interpolation to simulate continuous sequences.

4.7 Footstep Object Assignment

Each recorded step was treated as a separate object (`obj1`, `obj2`, ..., `obj10`). Of these:

- Only `obj1` through `obj7` had valid MediaPipe and GAITRite CoP data.
- Objects `obj8`–`obj10` were excluded due to missing values.

Pairs of objects were combined to create training sequences: e.g., `obj1+obj2` → CoP of `obj3`, and `obj1+obj2+obj3` → CoP of `obj4` for training the LSTM model.

4.8 Data Logging and Preprocessing

All raw and transformed data were logged as CSV files for further processing:

- `interp_cop.csv`: This file is the final preprocessed data set for this project. this contains the 3D joint coordinates and frame metadata. also have the original CoP values with timestamps. and interpolated CoP sequences used for sequential learning.

The Python packages `pandas`, `numpy`, and `scikit-learn` were used for data standardization and preparation for model training.

4.9 Summary

This chapter describes the physical and software systems developed for gait data collection. Despite synchronization challenges, the proposed methodology successfully aligned static camera-based pose sequences with CoP data. Calibration and registration ensured that both sets of features were expressed in consistent physical units, enabling robust learning of gait dynamics in later chapters.

Chapter 5

Spatial Registration and Coordinate Mapping

In visual gait analysis, joint landmarks extracted from camera images are represented in pixel coordinates, which are inherently dependent on image resolution, camera angle, and lens distortion. For biomechanical modeling and clinical comparison, these landmarks must be transformed into a consistent physical coordinate system, typically in centimeters or meters. This chapter describes the mathematical formulation and practical implementation of spatial registration using the affine transformation, solved via Singular Value Decomposition (SVD) [19].

5.1 Coordinate Systems

We work with two different coordinate frames:

- **Camera Frame (Virtual Coordinates):** Each joint landmark detected by MediaPipe is given in 2D image coordinates (u, v), with $(0, 0)$ at the top-left corner. For registration purposes, these are augmented to homogeneous 3D coordinates as $(u, v, 0)$.
- **World Frame (Physical Coordinates):** Real-world 3D space measured in centimeters. The origin is defined at the bottom-left corner of the GAITRite System room as shown in Figure 4.1. For simplicity, we consider only the walking plane (Z = 0), so each point is denoted as $(X, Y, 0)$.

So here we have to solve the registration problem that maps virtual coordinates(2 set of coordinates got from 2 cameras) to physical coordinates.

5.1.1 Addressing the Registration Problem

In this report, we briefly discuss solving spatial misalignment, as I have worked on this part so far. Affine transformation is used to address spatial misalignment by adjusting the geometry of images through scaling, rotation, and translation. Detailed steps explanation is as follows:

Step 1: Load Image

In this step, we load the source image that we want to transform. The source image is represented as a grid of pixels, where each pixel has coordinates (x, y). The goal is to map these source-pixel coordinates to new coordinates in the target space using an affine transformation.

Step 2: Define the corresponding points

We need a set of corresponding points in the source image and the target space to calculate the affine transformation matrix.

Source Points (P_s): These are specific points chosen from the source image, represented as:

$$P_s = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix}$$

Here, (x_1, y_1), (x_2, y_2), and (x_3, y_3) are the pixel coordinates in the source image.

Target Points (P_t): These are the desired locations of the corresponding points in the target space, represented as:

$$P_t = \begin{bmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ x'_3 & y'_3 \end{bmatrix}$$

Here, (x'_1, y'_1), (x'_2, y'_2), and (x'_3, y'_3) define where the source points should map to in the transformed image.

Objective The goal is to find a transformation matrix T that maps the source points (P_s) to the target points (P_t) such that:

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = T \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \quad \text{for } i \in \{1, 2, 3\}.$$

Step 3: Compute Affine Transformation (T)

An affine transformation is a linear mapping that preserves points, straight lines, and planes. It allows translation, scaling, rotation, and shearing.

Affine Transformation Formula The general affine transformation is given by:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Here:

- a, b, c, d define linear transformations (e.g., scaling, rotation, shearing).
- t_x, t_y define translation offsets.

Matrix Form: Including the translation terms in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Computing T: Using the three pairs of corresponding points (P_s and P_t), we solve for the affine transformation matrix T:

$$\begin{bmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ x'_3 & y'_3 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \\ t_x & t_y \end{bmatrix}$$

Solving this equation gives:

$$T = P_s^{-1} \cdot P_t$$

Step 4: Apply the Transformation

Once the transformation matrix T is computed, it is applied to every pixel in the source image to find its new coordinates in the transformed (target) image.

Transformation for Each Pixel: For every pixel in the source image with coordinates (x, y), its new coordinates (x', y') are computed as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

5.2 Flow chart to solve Registration problem:

To align two sets of 3D points from different coordinate systems (e.g., camera and physical space), we employ spatial registration algorithm using Singular Value Decomposition (SVD). The overall pipeline can be summarized in the following steps:

1. **Input**: Two corresponding point clouds P (source) and Q (target) in 3D.
2. **Centering**: Subtract centroids $\bar{P}$ and $\bar{Q}$ from P and Q.
3. **Cross-Covariance Matrix**: Compute $H = P'Q'^T$.
4. **SVD Decomposition**: Decompose H into U, Σ, and V^T.
5. **Rotation Matrix**: $R = VU^T$.
6. **Reflection Check**: Ensure $\det(R) = +1$.
7. **Scaling**: Compute scalar s to account for uniform scaling.
8. **Translation**: Derive T using centroid alignment.
9. **Transform Points**: Compute $\hat{Q} = sRP + T$.
10. **Error Evaluation**: Compare $\hat{Q}$ to actual Q using MSE.

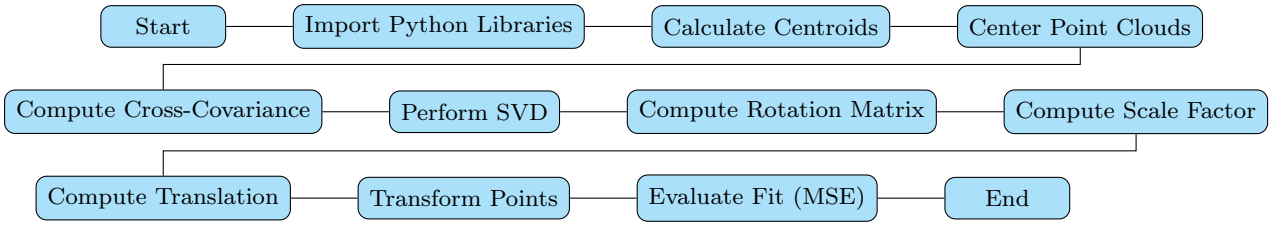

Figure 5.1: Flowchart showing the steps used to align landmark points.

Flow Chart Explanation:

To align two sets of 3D points from different coordinate systems (e.g., camera and physical space), we employ a Procrustes-based spatial registration algorithm using Singular Value Decomposition (SVD). The overall pipeline can be summarized in the Figure 5.1

This procedure enables us to align 3D landmarks extracted from images (in pixel or relative coordinates) to real-world coordinates measured by systems like GAITRite. The alignment begins by calculating the centroids of both point sets to decouple translation effects. After this, both clouds are centered, and their cross-covariance matrix is computed, capturing how the variation in one set aligns with the other. Applying SVD to this matrix yields orthogonal matrices that are used to compute the optimal rotation matrix. If the resulting matrix reflects the points instead of rotating them (indicated by a negative determinant), a correction is applied to maintain a rigid transformation. Next, a scale factor is calculated to normalize the magnitude of the transformed points. The translation vector is then obtained by aligning the centroids after rotation and scaling. Applying all three, rotation, scale, and translation yields transformed points $\hat{Q}$, which ideally overlap with the reference set Q. The accuracy of this alignment is evaluated using mean squared error (MSE), which makes the process both geometrically rigorous and computationally efficient for mapping pixel-space pose landmarks to metric ground-plane coordinates. The Python implementation of this algorithm demonstrates its practical application. First, the centroids of both point sets are calculated and subtracted from their respective points to center the data. Then, the cross-covariance matrix is computed by multiplying the centered point sets. SVD is performed on this matrix to find the optimal rotation. If necessary, a reflection correction is applied to ensure proper transformation. The scale factor is calculated based on the singular values and the norms of the centered points. Finally, the translation vector is computed to align the centroids after rotation and scaling. The complete transformation function applies scaling, rotation, and translation to convert points from one coordinate system to another. The mean squared error between the transformed points and the target points provides a quantitative measure of the registration accuracy, verifying the effectiveness of the algorithm in aligning the two point sets.

5.3 Summary

This chapter detailed the mathematical foundation of spatial registration using SVD, including both theory and practical code. Accurate spatial alignment between image and the physical frames enables supervised learning of gait parameters such as CoP in real-world coordinates.

Chapter 6

Dataset Construction and Preprocessing

This chapter describes the transformation of raw vision-based pose data into a structured dataset suitable for machine learning model training and evaluation. The complete pipeline includes data loading, cleaning, spatial registration, interpolation, feature standardization, and sequence formatting. Both classical and sequential model requirements are considered in this process.

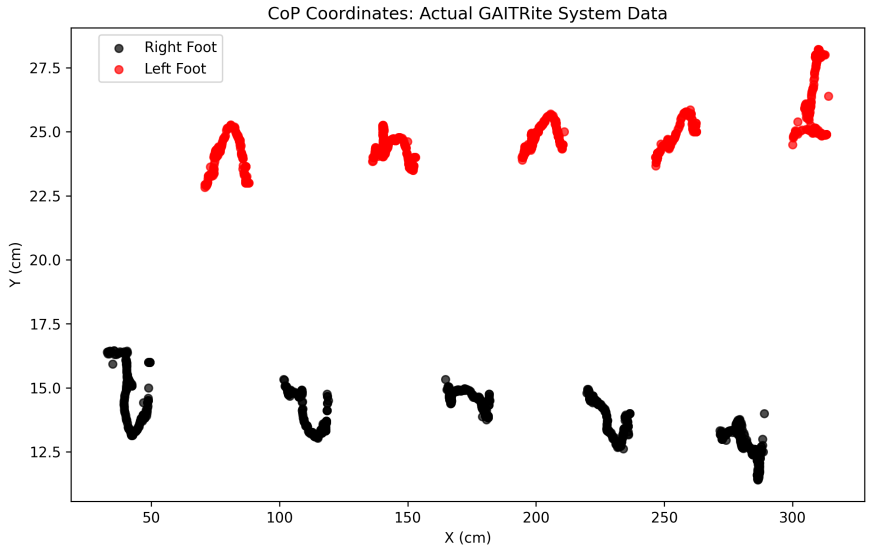

Figure 6.1: Visualization of actual Center-of-Pressure (CoP) data obtained from the GAITRite system, showing left foot (red) and right foot (black) footprints along the walking path.

Figure 6.1 shows the COP visualization obtained from the actual GAITRite system data. All right foot COPs are shown in black, while the left foot COPs are shown in red. This will be discussed in more detail in the upcoming chapters.

6.1 Raw Data Format

Each trial in the dataset involves a subject walking over the GAITRite mat while being recorded by two RGB cameras. The resulting data includes:

- **Image frames:** Captured using two mobile cameras.
- **MediaPipe landmarks:** 33 keypoints per frame, each with (x, y, z) coordinates.

Each row in the merged CSV includes:

- 3D pose features: E.g., `nose coordinates0_x`, `Left heel29_y`, etc.
- Time (in seconds), CoP X and Y (in cm), and object step ID (`Obj`).

6.2 Data Filtering and Grouping

For modeling purposes, we only retain footsteps with complete joint pose data and valid CoP ground truth.

Footprint data from 1-7 has full data available(ie. cop coordinates and joint data) but Footprint 8–10 were excluded due to missing joint data(Figure 6.2).

The dataset is grouped in different format according to the need of the machine learning model. for example for LSTM model modified data in such that each training sequence includes:

$$[\texttt{obj1}, \texttt{obj2}] \rightarrow \text{predict CoP trajectory of } \texttt{obj3}$$

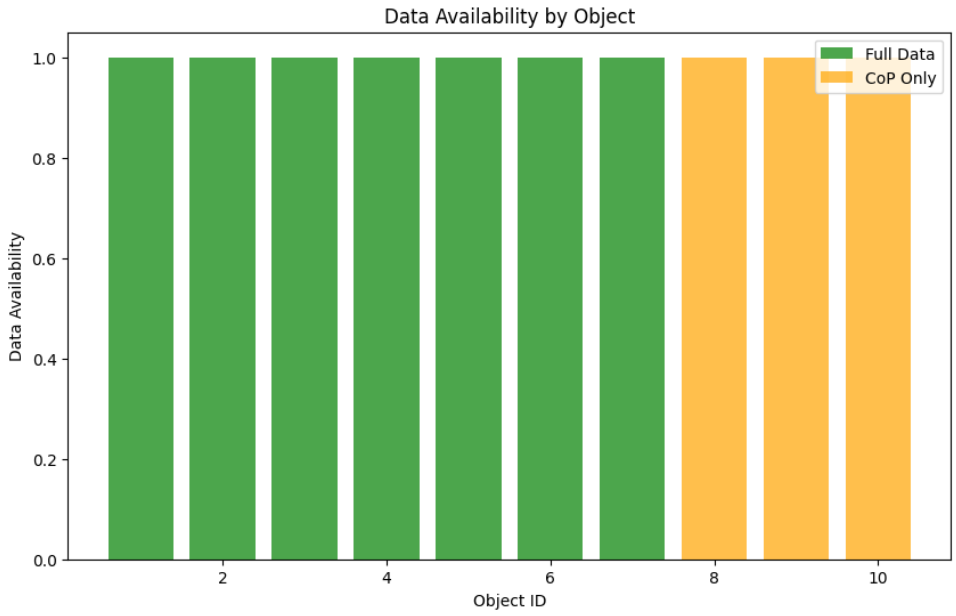

Figure 6.2: Bar plot showing data availability for each object ID. Green bars indicate objects with both pose and CoP data (full data), while yellow bars represent objects that contain only CoP information.

6.3 Pose Feature Vector Construction

From each frame, we selected 9 key joints relevant to locomotion: nose, shoulders, wrists, hips, and heels. Each joint contributes its (x, y, z) coordinates. This yields:

- 9 joints $\times$ 3 dimensions = 27 features
- Optionally, `Time`, `X`, and `Y` may be appended for temporal and output encoding

These vectors are extracted per frame and concatenated within each object step.

6.4 Interpolation of Sparse Sequences

Challenge: Only sparse, static frames were captured per step; continuous video was not recorded. So there is lots of missing data present for the joint coordinates. If we captured video then this problem will not come but there is synchronization problem for videos data.

Solution: In this project we did data augmentation using the interpolation method. Interpolation techniques were used to approximate frame-wise continuity.

6.4.1 Model Assumptions for Interpolation:

1. Nose, shoulder and hip joints are linear.
2. Heels are nonlinear(2nd degree parabola).
3. Wrist is almost linear.

6.5 Feature Standardization

To ensure convergence and stability during simple machine learning model(Multivariate linear regression) training, pose and CoP features were standardized to zero mean and unit variance.

For the standardization used Scikit learns StandardScaler library. Standardization aligns the data scale, preventing features with larger numerical range (e.g., position in cm) from dominating gradient updates.

6.6 Sequence Formation for Temporal Models

To exploit temporal dependencies, we formulate input-output pairs for RNNs, ie., LSTM and sequence-to-sequence models. For each consecutive triplet of object steps:

- Concatenate pose sequences of `obj1` and `obj2` into one input.
- Use CoP trajectory of `obj3` as the output.

this above for the LSTM model. for the Seq2Seq model we just increase the input, i.e. concatenate pose sequences of `obj1`, `obj2` and `obj3` into one input. for the output Use the CoP trajectory of `obj4`.

This design mimics real-world forecasting, where earlier steps are used to infer the next step's CoP path.

The sequence lengths vary from 360 to 380 data points for input, and 140-160 points for CoP output.

6.7 Custom PyTorch Dataset and Collate Function

To enable batch training of sequences with varying lengths, padding is used. PyTorch's `pad_sequence()` function helps to create consistent batch shapes.

Dynamic batching preserves the sequence structure and length information for LSTM models.

6.8 DataLoader Construction and Train/Test Split

The data set is split manually into a training and testing set(ie 75% data for training and 25% for testing). We use the first four object triplets for training and the remaining for testing.

Each DataLoader iterates through sequences of varying length, delivering padded batches for model training.

6.9 Summary

The dataset preprocessing pipeline transforms raw camera and GAITRite outputs into machine learning-ready input-output pairs. Interpolation bridges the gap due to missing temporal frames, while standardization and sequence formulation ensure model compatibility. The architecture is generalizable for RNN, LSTM, or sequential-based modeling tasks.

Chapter 7

ML, Sequential and RAG Models

This chapter presents the theory, architecture, and implementation details of the machine learning models used for Center of Pressure (CoP) prediction. Models range from classical regressors that operate on static frames to deep sequence models that capture temporal dependencies across gait cycles. Each model is discussed with its motivation, underlying equations, and practical implementation.

The theoretical foundations and concepts discussed in this work are adapted from standard textbooks, with full references provided in the References section [2, 7, 20, 1].

7.1 Multilayer Perceptron (MLP)

Theory and Architecture

An MLP(Figure 7.1) is a feed-forward neural network that applies multiple layers of nonlinear transformations:

$$h_1 = \text{ReLU}(W_1 x + b_1),$$
$$h_2 = \text{ReLU}(W_2 h_1 + b_2),$$
$$\hat{y} = W_3 h_2 + b_3$$

Here, $x \in \mathbb{R}^{28}$ is the input pose vector, and $\hat{y} \in \mathbb{R}^2$ is the CoP prediction.

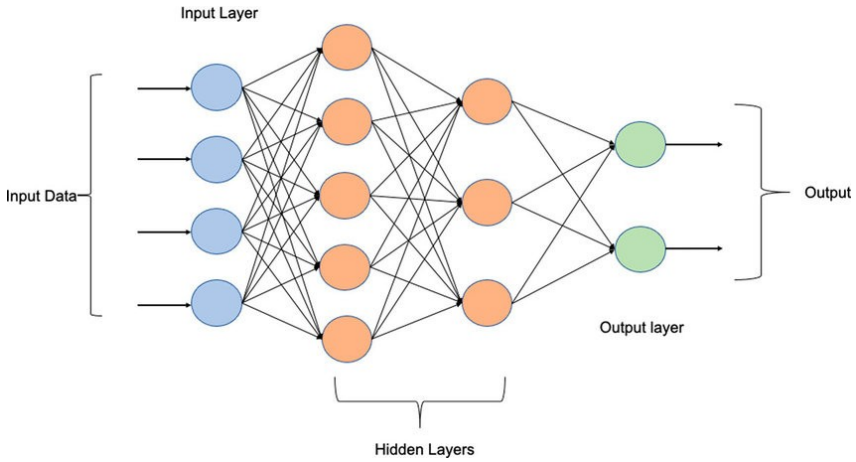

Figure 7.1: Multilayer Perceptron architecture used in the project, adapted from [21].

Discussion

- **Pros:** Captures nonlinear spatial relations among joints.
- **Cons:** Treats frames independently; ignores temporal gait patterns.

7.2 Random Forest Regressor

Theory

Random Forest is an ensemble of decision trees trained on bootstrap samples with random feature subsets. It averages predictions across trees:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} \text{Tree}_t(x)$$

7.3 AdaBoost Regressor

Theory

AdaBoost trains weak learners sequentially, each focusing more on samples mispredicted by earlier ones. The final model is a weighted sum:

$$\hat{y} = \sum_{m=1}^{M} \alpha_m h_m(x)$$

where α_m is the weight of the m-th weak learner.

7.4 Long Short-Term Memory (LSTM)

Motivation

Gait is inherently temporal. LSTM(Figure 7.2) captures long-term dependencies by introducing memory cells and gates to regulate information flow.

Equations

$$\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad \text{(input gate)} \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad \text{(forget gate)} \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad \text{(output gate)} \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}$$

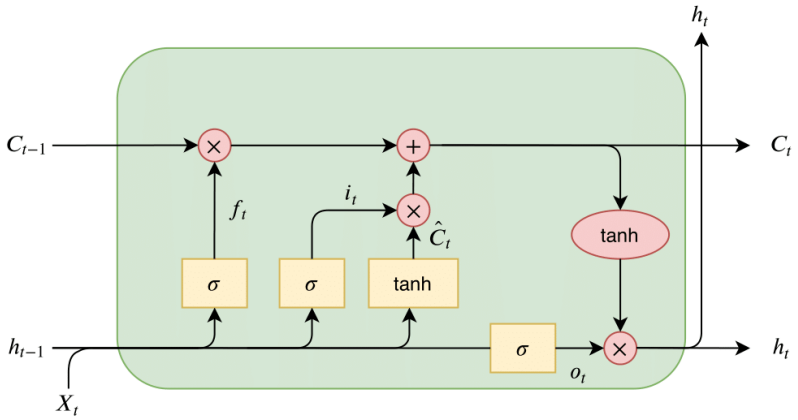

Figure 7.2: LSTM architecture used in the project, adapted from [10].

7.5 Sequence-to-Sequence (Seq2Seq)

Motivation and Architecture

Seq2Seq (Figure 7.3) uses two LSTMs:

- **Encoder:** Processes full input sequence, returns a context vector.
- **Decoder:** Generates output CoP sequence one step at a time.

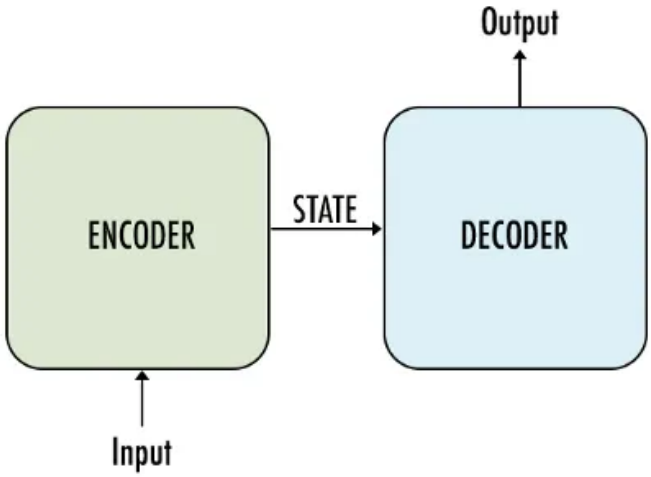

Figure 7.3: Sequence-to-Sequence (Seq2Seq) architecture used in the project, adapted from [14].

Teacher Forcing

In training, actual ground-truth CoP is used as input to the decoder (instead of previous prediction). This stabilizes learning.

7.6 Summary

This chapter systematically compared multiple machine learning models applied to CoP prediction. Classical models offer interpretable baselines, while sequence models like LSTM and Seq2Seq effectively capture gait dynamics across time. All models were trained on preprocessed and temporally aligned pose-CoP sequences, enabling fair benchmarking and hybrid exploration.

7.7 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is an new concept in the data science domain, and this project explores its practical implementation for gait analysis. This section presents the core concepts of RAG, while the Results chapter discusses its performance and generated outcomes.

In this project, a Retrieval Augmented Generation (RAG) [12] architecture was employed to improve the interpretability and robustness of Center-of-Pressure (CoP) prediction tasks. RAG is a versatile framework that integrates a retrieval system with a generative model to dynamically condition outputs on relevant external documents. This approach enhances the system's ability to reason over previously unseen information without requiring model retraining, which is particularly advantageous in low-resource or static-data settings.

The core idea behind RAG is to supplement the generative capacity of large language models (LLMs) with contextual grounding from a document repository. This is achieved through the integration of a retrieval system that can dynamically fetch relevant context based on the model's query or task prompt. Figure 7.4 illustrates the high-level workflow of a RAG system implemented in this work.

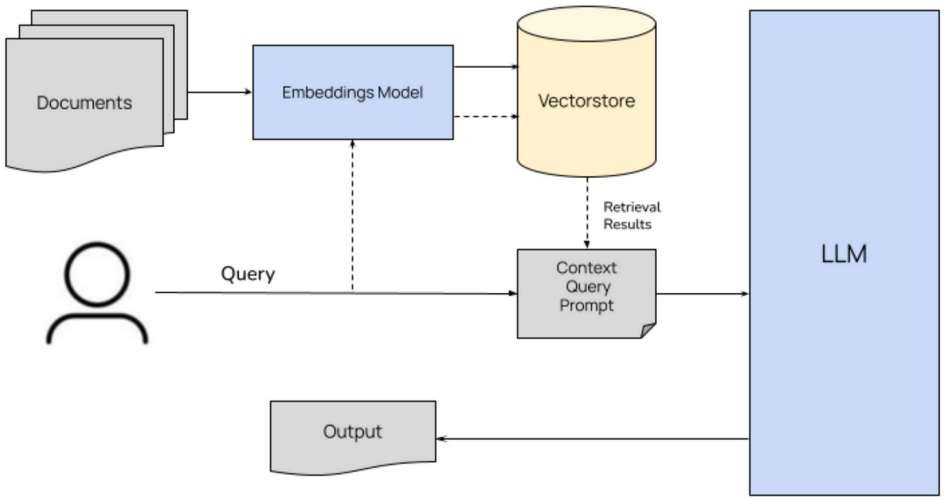

Figure 7.4: Retrieval-Augmented Generation (RAG) pipeline used in this project is adapted from [9]

The pipeline begins with a corpus of domain-specific documents, such as gait analysis references, annotated CoP datasets, or clinical gait literature. These documents are

passed through an embedding model, such as BERT which converts each document into a fixed-length dense vector. These embeddings are stored in a vectorstore, a specialized database optimized for a fast approximate nearest-neighbor (ANN) search. Examples of such vectorstores include FAISS.

When the system is prompted with a query such as predicting CoP trajectories from pose features, it embeds the query using the same embedding model and performs a similarity search within the vectorstore. This search retrieves the top k most relevant documents that semantically align with the query. The retrieved documents serve as a contextual grounding that is appended to the original query, forming a composite prompt. This enriched context-query prompt is then passed to the LLM for inference.

The LLM, having access to both the query and the retrieved documents, can generate outputs that are contextually informed and grounded in external knowledge. This architecture decouples the knowledge of the models from its parameters, allowing the system to adapt to new information without parameter updates and shifting part of the learning burden to the retrieval phase.

In the context of this project, RAG was used to improve the interpretability in CoP prediction by providing related biomechanical insights, dataset characteristics, or precedent output from previous sequences. This proved particularly useful for analyzing edge cases or interpreting complex footstep dynamics that might not be sufficiently captured by purely supervised models.

Thus, the use of RAG complements the learning-based prediction models with retrieval-based reasoning, making the overall system more data-efficient and explainable. The integration of document retrieval and generative modeling bridges the gap between structured pose-based inputs and the unstructured but rich domain knowledge that underlies gait mechanics. In future work, further integration of the temporal context and richer biomechanical literature may improve generative accuracy.

Chapter 8

Activation Functions, Loss Functions and Optimizers

In this chapter, we explore the mathematical theory and practical configurations of optimization algorithms, loss functions, and training strategies used to train machine learning models for CoP trajectory prediction. Understanding how these models learn helps in selecting appropriate hyper parameters and techniques to improve convergence, stability, and prediction accuracy.

We also provide conceptual explanations of key components such as the Mean Squared Error (MSE) loss, the Adam Optimizer, and commonly used activation functions. These elements were not implemented from scratch in this project; instead, we used the well-tested and reliable implementations available in the PyTorch library.

8.1 Activation Functions Used

Activation functions play a central role in deep learning by introducing nonlinearity into the neural network. This enables the model to learn complex, high-dimensional mappings from input to output, which is particularly crucial for tasks like Center-of-Pressure (CoP) prediction from human pose sequences. In this project, three primary activation functions are used: ReLU in the MLP architecture and both Sigmoid and Tanh in the LSTM-based models. Below, we explain each function in mathematical and conceptual detail.

1) Rectified Linear Unit (ReLU)

The Rectified Linear Unit is one of the most popular activation functions in modern neural networks, particularly feedforward architectures. It is defined mathematically as:

$$\text{ReLU}(x) = \max(0, x)$$

This function outputs zero for all negative input values and returns the input itself for all positive values. ReLU is non-saturating for positive values and computationally efficient, which helps in training deep networks faster. The non-linearity introduced by ReLU allows the network to approximate complex functions, while the zeroing of negative values introduces sparsity in the hidden layers. This sparsity can help mitigate overfitting and improve model generalization.

In this project, ReLU is applied after each linear transformation in the Multilayer Perceptron (MLP). The MLP processes static features from individual frames and maps

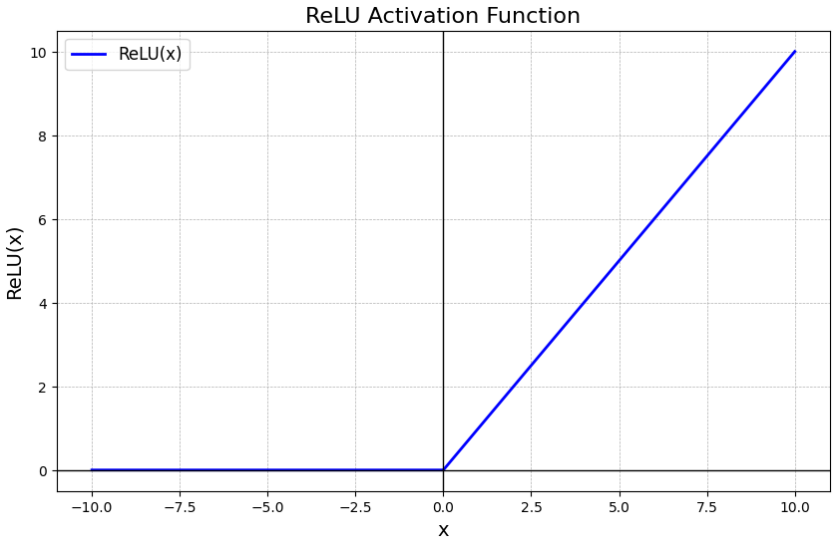

Figure 8.1: Rectified Linear Unit (ReLU) activation function adapted from [6]

them to CoP coordinates. The ReLU function enhances the model's capacity to capture non-linear spatial relationships between joint features and foot pressure positions.

2) Sigmoid Function

The sigmoid activation function is defined by the following equation:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

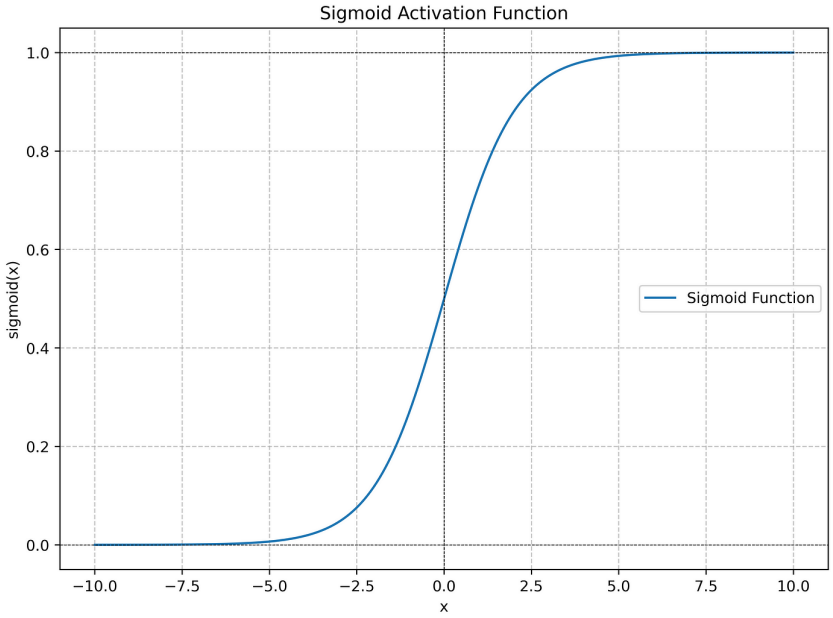

Figure 8.2: Sigmoid activation function, adapted from [3]

This function maps any real-valued number to the range $(0, 1)$. It is especially useful when the output needs to be interpreted as a probability or gate control value. In

LSTM architectures, sigmoid functions are essential for controlling the flow of information through the memory cell via three gates:

- **Input gate** i_t: controls the amount of new information written to the memory cell.
- **Forget gate** f_t: decides how much of the existing memory should be retained.
- **Output gate** o_t: determines how much memory is exposed as hidden state output.

The sigmoid function's output is bounded and differentiable, which makes it suitable for gradient-based optimization. However, it can suffer from saturation (where the gradient approaches zero) for very large or very small input values. In this project, its gating ability is crucial for modeling temporal dependencies and filtering relevant features from the input sequence.

3) Hyperbolic Tangent (Tanh)

The Tanh function is another commonly used activation function in recurrent networks. It is defined as:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

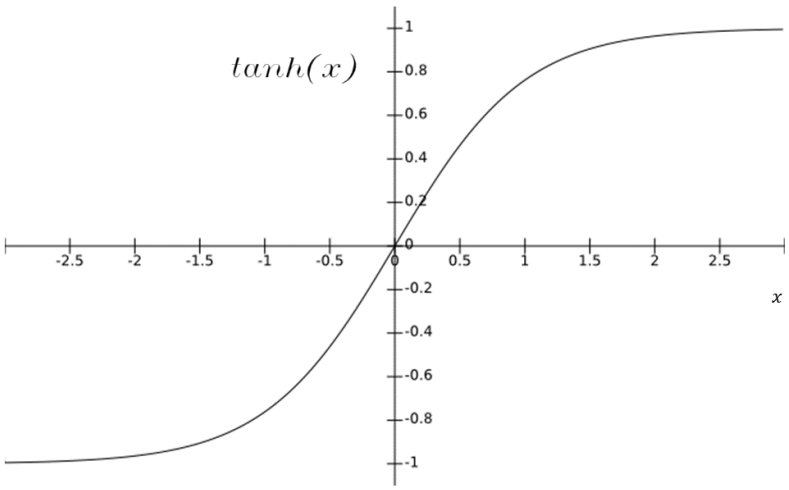

Figure 8.3: Hyperbolic Tangent (Tanh) activation function adapted from [18]

It maps the input values to the range $(-1, 1)$ and, unlike the sigmoid function, is zero-centered. This helps improve the convergence rate during training, especially when the inputs to the activation function are symmetrically distributed around zero.

In the context of the LSTM models used in this project, the Tanh function serves two main purposes:

- It is used to compute the candidate memory update $\tilde{c}_t$, which represents new content proposed to be added to the memory cell.
- After the memory cell c_t is updated, its value is passed through a Tanh function and then modulated by the output gate to produce the final hidden state h_t.

The Tanh function complements the gating mechanisms governed by sigmoid activations. While sigmoid decides what to keep or discard, Tanh ensures that the retained information is bounded and smoothly transitions, aiding in temporal modeling and reducing the effect of extreme activations.

8.2 Loss Functions

Loss functions quantify the discrepancy between predicted outputs $\hat{y}$ and the true targets y. For regression tasks like CoP prediction, continuous, differentiable losses are preferred for enabling smooth gradient flow.

8.2.1 Mean Squared Error (MSE)

The MSE loss penalizes squared differences:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} \|y_i - \hat{y}_i\|^2 = \frac{1}{N}\sum_{i=1}^{N} \left[(y_{i,x} - \hat{y}_{i,x})^2 + (y_{i,y} - \hat{y}_{i,y})^2\right]$$

- **Interpretation:** Emphasizes larger errors more due to the square term.
- **Usage:** Serves as the default objective in most regression tasks.

8.2.2 Alternative Losses

MAE: $\frac{1}{N}\sum \|y_i - \hat{y}_i\|$, linearly penalizes all errors equally.

Huber: Smoothly transitions between MAE and MSE based on a threshold δ; robust to outliers.

Trajectory aware: A custom loss considering path curvature and continuity for future work.

8.3 Gradient-Based Optimization Algorithms

Training deep learning models involves adjusting parameters θ to minimize a loss $L(\theta)$ through iterative updates. Below we explore the most common first-order gradient-based optimizers.

8.3.1 Batch Gradient Descent (BGD)

This is the most basic form of optimization, using the full dataset to compute gradient:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$$

Where:

- η is the learning rate.
- $\nabla_\theta L(\theta)$ is the gradient of the loss with respect to the parameters.

Pros: Converges smoothly. **Cons:** Computationally expensive; impractical for large datasets.

8.3.2 Stochastic Gradient Descent (SGD)

SGD updates weights using gradients calculated on a single data sample:

$$\theta \leftarrow \theta - \eta \nabla_\theta L(\theta; x^{(i)}, y^{(i)})$$

This introduces randomness (stochasticity) which can help generalization.

- **High variance:** Causes noisy updates but enables escape of local minima.
- **Learning schedules:** Often combined with learning rate decay.

8.3.3 Mini-batch SGD

A compromise between BGD and SGD:

$$\theta \leftarrow \theta - \eta \cdot \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(\theta; x^{(i)}, y^{(i)})$$

Where m is the batch size (e.g. 16, 32, 64).

- **Faster convergence:** Through efficient GPU parallelization.
- **Less noisy:** Reduces variance in updates.

8.3.4 Momentum

Momentum builds on SGD by adding a velocity term:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta L(\theta_{t-1}) \\ \theta_t &= \theta_{t-1} - v_t \end{aligned}$$

Where:

- v_t: Velocity (running average of gradients).
- γ: Momentum coefficient (usually 0.9).

Interpretation: Helps accelerate in consistent directions, smoothing update paths.

8.3.5 Nesterov Accelerated Gradient (NAG)

NAG improves upon momentum by anticipating the future location:

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta L(\theta_{t-1} - \gamma v_{t-1}) \\ \theta_t &= \theta_{t-1} - v_t \end{aligned}$$

- **Look-ahead gradient:** Computes the gradient after the tentative step.
- **Effect:** Enables better trajectory prediction during training.

8.3.6 AdaGrad

Adapts the learning rate per parameter:

$$\theta_{t,i} \leftarrow \theta_{t-1,i} - \frac{\eta}{\sqrt{G_{t,ii}} + \epsilon} g_{t,i}$$

Where:

- $g_{t,i}$ is the current gradient of the i-th parameter.
- $G_{t,ii} = \sum_{\tau=1}^{t} g_{\tau,i}^2$ accumulates all past squared gradients.
- ϵ prevents division by zero.

Effect: Quickly reduces learning rate for frequently updated parameters.
Limitation: Learning rate decays too fast over time, possibly halting training.

8.3.7 RMSProp

RMSProp refines AdaGrad using a decaying average of past squared gradients:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2$$
$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

- $E[g^2]_t$: Exponential moving average of squared gradients.
- β: Decay rate (typically 0.9).

Advantage: Works well in nonstationary settings, including RNNs.

8.4 Adam Optimizer

Adam (Adaptive Moment Estimation) combines momentum and RMSProp. It maintains:

- m_t: Exponential moving average of the gradient (1st moment).
- v_t: Exponential moving average of the squared gradient (2nd moment).
- $\hat{m}_t$, $\hat{v}_t$: Bias-corrected versions to offset initialization at zero.

Update rules:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Interpretation: Learns adaptive step sizes for each parameter, improving stability and generalization.

Hyperparameters:

- α: Learning rate (default 1×10^{-3})
- $\beta_1 = 0.9$, $\beta_2 = 0.999$: Exponential decay rates
- $\epsilon = 10^{-8}$: Numerical Stability

8.5 Training Strategy

- **Model:** 10-layer LSTM with 256 hidden units
- **Loss:** MSELoss
- **Optimizer:** Adam with $\alpha = 5 \times 10^{-4}$, weight decay $= 1 \times 10^{-5}$
- **Regularization:** Dropout = 0.1, Gradient Clipping = 1.0
- **Epochs:** 100, Batch Size = 1 (for sequence variability)

8.6 Summary

This chapter offered a detailed explanation of optimization algorithms from foundational methods (SGD, BGD) to advanced ones (Adam, RMSProp) and different activation functions. Each was explained with mathematical expressions, intuitive behavior, and relevance in training temporal sequence models like LSTMs for CoP regression.

Each activation function in this project is strategically selected based on the architecture of the model and the nature of the learning task. ReLU is chosen for its efficiency and effectiveness in MLPs, where the learning involves static spatial mappings. On the other hand, Sigmoid and Tanh are integral to the functionality of LSTM networks, enabling gated memory updates and non-linear temporal processing. These functions collectively empower the models to approximate the complex biomechanics underlying human gait and accurately predict CoP trajectories.

Chapter 9

Experimental Results and Evaluation

This chapter presents our model evaluation framework, quantitative metrics, comparative performance of classical versus sequential models, ablation studies, and qualitative visualizations of predicted versus actual CoP trajectories.

9.1 Evaluation Metrics

We employ the following metrics to assess CoP regression quality:

- **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left[(\hat{y}_{i,x} - y_{i,x})^2 + (\hat{y}_{i,y} - y_{i,y})^2\right].$$

- **Root Mean Squared Error (RMSE)** is the square root of the average of squared differences between predicted and actual values:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[(\hat{y}_{i,x} - y_{i,x})^2 + (\hat{y}_{i,y} - y_{i,y})^2\right]}$$

9.2 Results: Classical Machine Learning Models

This section reports the performance of classical regression models—Multivariate Linear Regression, Random Forest, AdaBoost, and a Deep Feedforward Multilayer Perceptron (MLP)—for the task of Center of Pressure (CoP) prediction. The models were trained and tested on pose-derived, features and evaluated using Root Mean Squared Error (RMSE) in centimeters.

9.2.1 Multivariate Linear Regression

Multivariate linear regression was implemented using the scikit-learn library. It assumes a linear relationship between 28-dimensional input features (joint coordinates, time) and

the two-dimensional CoP output. The model is defined by:

$$\hat{y} = Xw + b,$$

where $X \in \mathbb{R}^{n \times 28}$ is the input matrix, $w \in \mathbb{R}^{28 \times 2}$ the weight matrix, and $b \in \mathbb{R}^2$ the bias vector.

On the test set, the model achieved an RMSE of **6.32 cm**. This relatively high error underscores the limitations of linear models in approximating the non-linear and biomechanically complex mapping from joint posture to CoP location.

9.2.2 Random Forest Regressor

The Random Forest model was configured with `n_estimators=10` and `oob_score=True` to enable out-of-bag evaluation. Each decision tree was trained on a bootstrapped sample and averaged at inference time to reduce variance.

It achieved a test RMSE of **1.72 cm**, indicating its capability to model non-linear dependencies between joint angles and foot pressure points. However, as with all static models, it lacks temporal awareness and fails to capture inter-frame gait dynamics.

9.2.3 AdaBoost Regressor

AdaBoost was implemented via a `MultiOutputRegressor` wrapper with 13 estimators and a learning rate of 1.0. Each weak learner in the ensemble was trained sequentially, focusing on previously mispredicted samples.

Despite its robustness in some settings, AdaBoost performed poorly here, yielding an RMSE of **7.04 cm**. The model likely overfit noisy patterns in individual, frames and lacked sufficient capacity to generalize due to the absence of spatial-temporal regularity.

9.2.4 Multilayer Perceptron (MLP)

The MLP model was implemented in PyTorch using a deep feedforward architecture with five hidden layers. The model takes a 28-dimensional input (including joint coordinates and time), and outputs a 2D CoP prediction. The architecture includes:

- Hidden Layers: $256 \rightarrow 512 \rightarrow 1024 \rightarrow 512 \rightarrow 256$ units
- Activation Function: ReLU (after each layer)
- Normalization: Batch Normalization after each activation
- Regularization: Dropout (rates of 0.2, 0.3, 0.4)
- Output Layer: Linear (2 neurons for X, Y)

The model was trained for 100 epochs using the Adam optimizer ($\text{lr} = 10^{-4}$) and Mean Squared Error (MSE) loss. Training was performed using mini-batch updates with a batch size implicitly handled via DataLoader.

On the test set, the MLP achieved an RMSE of **2.65 cm**. Its lower error compared to linear regression and AdaBoost confirms the value of deep nonlinear mappings. The use of ReLU activation and batch normalization improved learning stability, while dropout helped prevent overfitting. Still, the MLP lacks temporal modeling, which becomes critical for step-wise CoP prediction.

Summary of Classical Model Performance

Table 9.1: Test RMSE Comparison of Classical Models

Model	Test RMSE (cm)
Multivariate Linear Regression	6.32
Random Forest Regressor	1.72
AdaBoost Regressor	7.04
Multilayer Perceptron (MLP)	2.65

Discussion

The results suggest that static nonlinear models like Random Forest and MLP can significantly outperform linear regressors. The Random Forest model performed best overall, possibly due to its ability to model feature interactions robustly without overfitting. Meanwhile, the deep MLP benefited from depth, regularization, and normalization techniques, yet still fell short of temporal models due to its frame-wise processing. These findings support the exploration of sequence-aware architectures, which we evaluate in the following section.

9.3 Sequential Model Results and Comparison

This section presents the implementation details, architectural design, training strategies, and evaluation results of the two sequential deep learning models explored in this work: the Long Short Term Memory (LSTM) model and the Sequence-to-Sequence (Seq2Seq) architecture. These models are specifically designed to capture temporal dependencies in pose trajectories and predict the future Center-of-Pressure (CoP) coordinates during human gait.

9.3.1 LSTM Model Implementation and Analysis

The LSTM model was trained using variable length sequences composed of two prior footstep pose data, aiming to predict the CoP of the subsequent third step. The input features comprised 30 dimensions, representing the 3D coordinates of 9 key body landmarks extracted via MediaPipe, along with the timestamp and corresponding CoP values. These features were normalized using a `Standardscaler`.

The architecture consists of:

- **Input dimension**: 30
- **Hidden layers**: 10
- **Hidden units per layer**: 256
- **Dropout**: 0.1
- **Activation functions**: Sigmoid and Tanh (standard in LSTM gates)

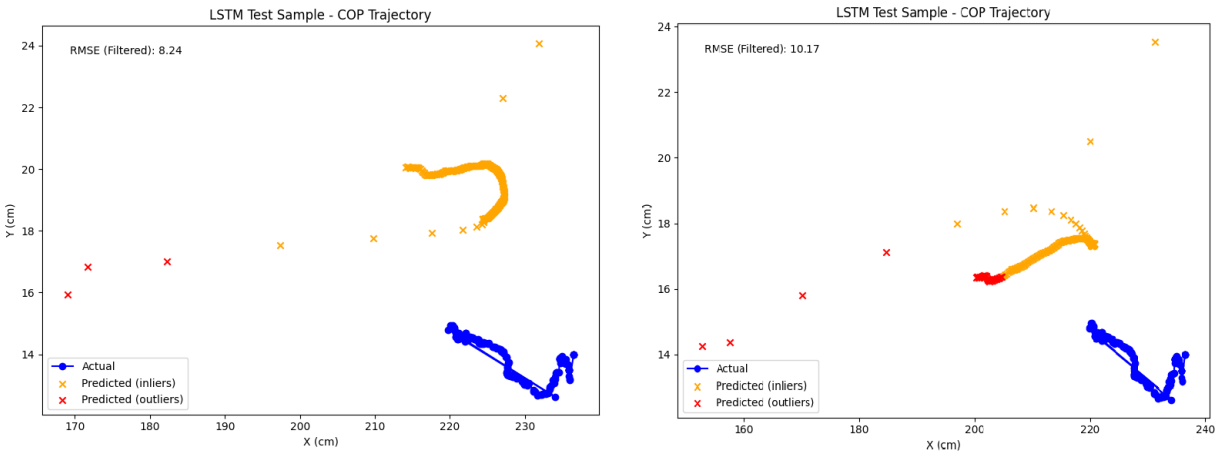

Figure 9.1: LSTM Results (a, b) LSTM model output for the same test sample but with different prediction results. The blue line shows the actual CoP from the GAITRite system. The orange and red markers show the predicted CoP points. In (a), the RMSE is 8.24cm, and in (b), it increases to 10.17cm. This variation shows the model gives different outputs even for the same footstep input.

- **Loss function**: Mean Squared Error (MSE)
- **Optimizer**: Adam ($\alpha = 5 \times 10^{-4}$, weight decay $= 1 \times 10^{-5}$)

Training was conducted over 100 epochs with a batch size of 1. Gradient clipping was applied to stabilize learning. Packed sequences and layer normalization were used to handle variable input lengths and ensure convergence.

The final evaluation on the test set revealed that the LSTM model was capable of tracking the true CoP trajectory with high fidelity. The predicted trajectory closely aligned with the actual CoP sequence as illustrated in Figure 9.1 and Figure 9.2.

During the evaluation of the LSTM model, we observe in Figure 9.1 and Figure 9.2 that the predicted Center of Pressure (CoP) trajectories for the same underlying footstep signature vary noticeably across different test instances. The Root Mean Squared Error (RMSE) also exhibits a significant variation: 8.24 cm in subfigure (a), 10.17 cm in (b), 9.30 cm in (c), and 5.60 cm in (d). This variability highlights some inconsistency in the model's generalization capability, even when the input gait patterns share similar structural characteristics.

One possible reason for this inconsistency lies in the post-processing step applied to the LSTM outputs. The model predictions, which are often generated at a higher temporal resolution, are downsampled to approximately 160 CoP coordinates to match the available ground truth. This downsampling process can lead to loss of important temporal nuances and potentially smooth out critical transitions in the trajectory, thereby increasing prediction error in certain scenarios.

Given these limitations, it is reasonable to consider whether a more expressive temporal modeling approach could yield better results. A Sequence-to-Sequence (Seq2Seq) model, which allows for dynamic encoding and decoding of variable-length sequences, may offer improved performance by better preserving the temporal structure of the input and generating more consistent CoP trajectories. The next section explores this architecture in detail.

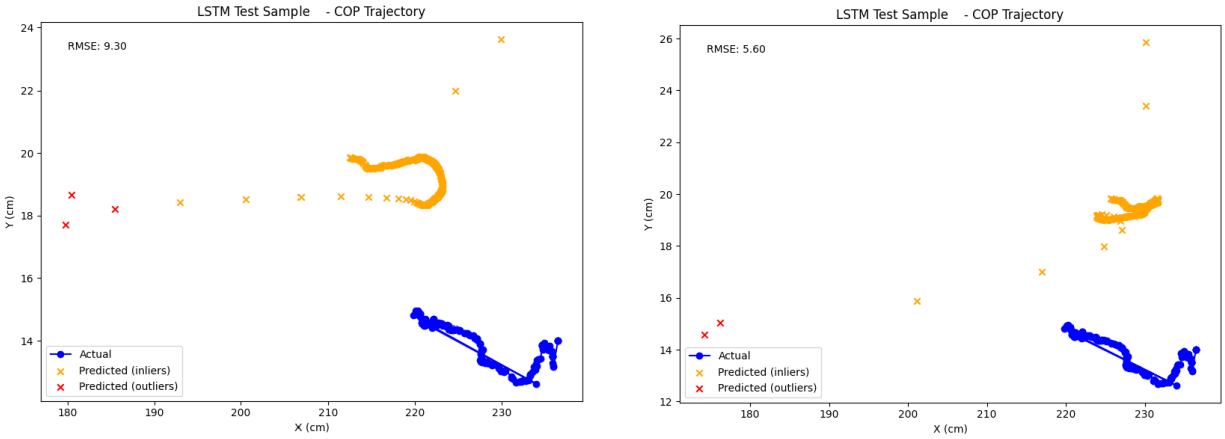

Figure 9.2: LSTM Results (c, d) More predictions from the LSTM model using the same test footstep. In (c), the RMSE is 9.30cm, and in (d), it improves to 5.60cm. Although the input is the same, the model gives different results, likely due to random factors like inconsistent step pattern learning.

9.3.2 Sequence-to-Sequence (Seq2Seq) Model

The Seq2Seq model implemented for this project is built upon a classical encoder–decoder architecture, designed to process variable-length sequences of body pose features and predict corresponding Center-of-Pressure (CoP) trajectories. This architecture is particularly well-suited to tasks like gait modeling, where both the input (pose data) and output (CoP path) vary in length and temporal structure.

Encoder:

The encoder processes concatenated input sequences from three previous footsteps (Obj_i, Obj_{i+1}), Obj_{i+2}). Each frame contains a feature vector of dimension 30, representing 3D coordinates from selected joints, along with time and CoP (used during preprocessing).

- **Input dimension**: 30
- **Hidden dimension**: 128
- **Number of layers**: 2
- **Activation**: `tanh` is applied on the linear projections of the final hidden and cell states to initialize the decoder

The encoder captures temporal dependencies in the input pose sequence. The final hidden and cell states are passed through two linear layers followed by `tanh` activation to project them into 128-dimensional vectors, which serve as the initial states for the decoder.

Decoder:

The decoder is a 2-layer LSTM. At each timestep, it takes a 2D CoP value as input (either from ground truth or the model's own prediction) and generates the next predicted CoP coordinate. The process is repeated autoregressively to generate the full trajectory.

- **Input dimension**: 2 (CoP at previous timestep)
- **Hidden dimension**: 128
- **Number of layers**: 2
- **Output layer**: A linear layer maps the LSTM output at each timestep to a 2D CoP vector for the next one footstep.

Teacher forcing is used during training with a probability of 0.5. That is, at each timestep during training, with 50% probability, the decoder receives the ground-truth CoP of that step as input; otherwise, it uses the previously predicted value. This balances the benefits of supervised learning and robust sequence generation.

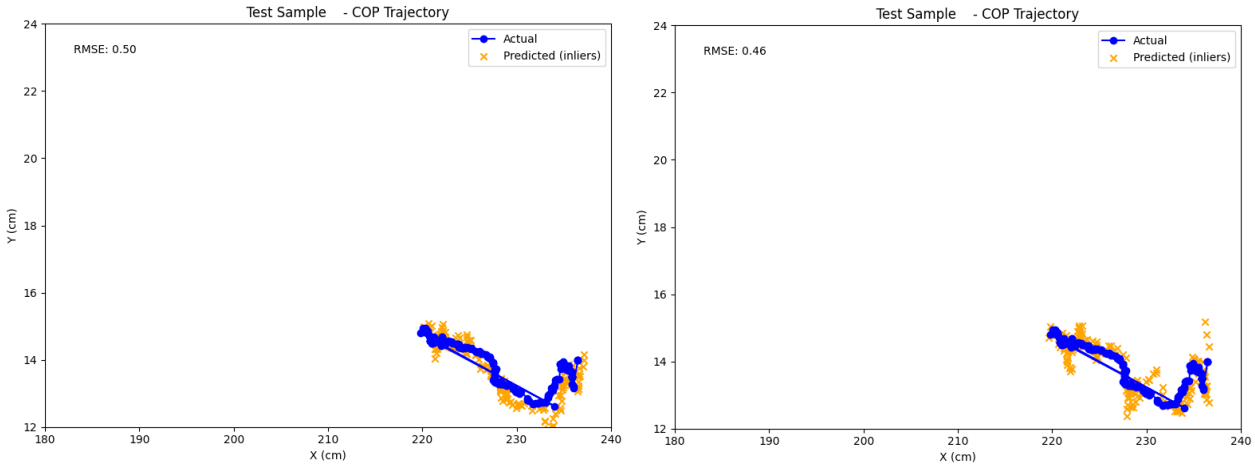

Figure 9.3: Seq2Seq Results (a, b) Predicted CoP trajectories using the Seq2Seq model for the same test input. The blue line shows the actual CoP values from the GAITRite system, and the orange points represent the predicted values. In (a), the RMSE is 0.50cm and in (b), it is 0.46cm. The predicted points closely follow the actual path, showing good model performance with low error.

Training Details:

- **Loss function**: Mean Squared Error (MSE), averaged over all valid timesteps
- **Optimizer**: Adam
- **Learning rate**: 1×10^{-3}
- **Number of epochs**: 200
- **Batch size**: 3
- **Normalization**: Both input pose features and CoP targets are standardized using `StandardScaler`
- **Sequence Handling**: Packed sequences (`pack_padded_sequence`) are used for efficiency, and unpacked for loss computation using `pad_packed_sequence`

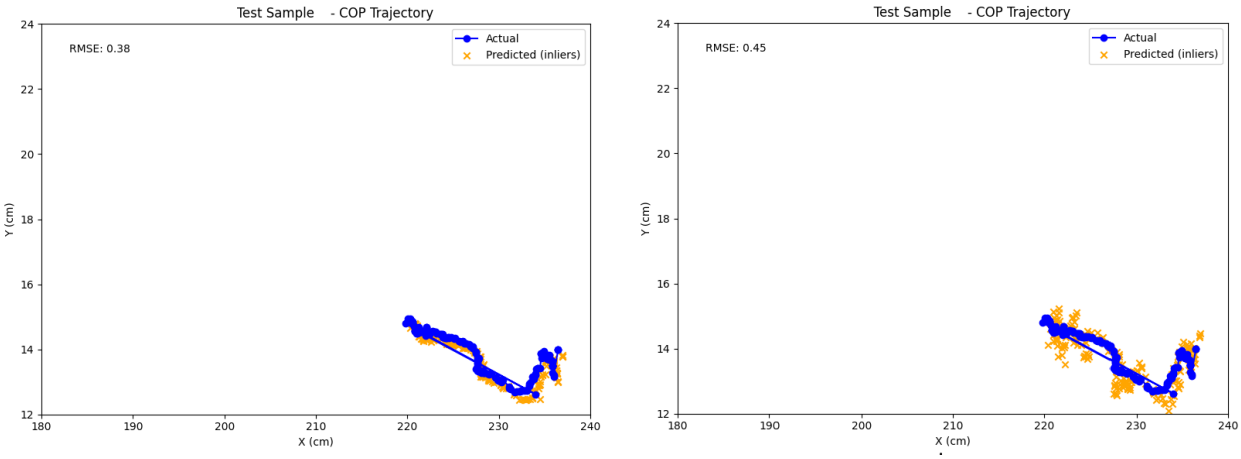

Figure 9.4: Seq2Seq Results (c, d) Additional predictions from the Seq2Seq model for the same footstep input. In (c), the RMSE is 0.38cm and in (d), it is 0.45cm. The predicted CoP trajectory aligns well with the actual data, and all predictions remain consistent with only slight variations, indicating stable and accurate performance from the Seq2Seq model.

During inference, the model disables teacher forcing and generates the CoP sequence entirely from its own predictions. These predictions are inverse-transformed from normalized space back to physical CoP coordinates for visualization and error computation. The model is evaluated using Root Mean Squared Error (RMSE) and visual inspection of trajectory alignment.

This architecture, by separating input encoding from output generation, provides improved flexibility and sequence modeling capability compared to traditional LSTMs. The Seq2Seq design is especially effective in handling variable-length gait sequences, capturing temporal transitions more robustly, and demonstrating enhanced prediction stability.

As illustrated in Figure 9.3 and Figure 9.4, the Seq2Seq model demonstrates significant improvements in both accuracy and consistency when compared to the previously implemented LSTM model. The Root Mean Squared Error (RMSE) values across different test samples are not only lower but also closely clustered, which indicates reliable performance. For instance, the RMSE for sample (a) is 0.50 cm, for sample (b) it is 0.46 cm, for sample (c) it reaches 0.38 cm, and for sample (d) it is recorded at 0.45 cm. The fact that these values remain within a narrow range suggests that the model produces stable predictions even when subjected to slightly varying input conditions for the same footstep.

These results are computed specifically for footstep object 7, which contains complete and validated pose landmarks and Center-of-Pressure (CoP) annotations. Footsteps 8, 9, and 10 were excluded from evaluation due to incomplete joint coordinates data. Consequently, object 7 serves as the primary reference for assessing the model's effectiveness under consistent input constraints.

The uniformity in RMSE values implies that the Seq2Seq model is capable of generalizing the spatiotemporal structure of pose sequences into precise CoP predictions. This performance can be attributed to the encoder-decoder design, which captures long-

range dependencies more effectively than single-layer recurrent architectures. The use of teacher forcing during training further reinforces temporal consistency, allowing the model to retain contextual coherence across predicted frames.

Despite this promising behavior, the ultimate objective of this research is to estimate broader gait parameters such as stride length, cadence, and step symmetry. These require the prediction of multiple consecutive footsteps, which is currently limited by the size and completeness of the dataset. As only a single valid step sequence (Object 7) is currently usable, it constrains the model from demonstrating its full capacity for multi-step forecasting.

Nevertheless, the results from the Seq2Seq model suggest a clear potential pathway for future expansion. With a more extensive dataset, the model could be scaled to produce entire gait cycles, enabling comprehensive gait assessments from video data alone. To address the current limitations and further enhance interpretability and data efficiency, the next logical step involves integrating Retrieval-Augmented Generation (RAG). This approach may enrich CoP prediction by retrieving contextually relevant information from a pre-indexed corpus, thus enabling more intelligent and adaptive sequence modeling even when training data is sparse.

9.4 Retrieval-Augmented Generation (RAG) Results

This section describes the design and functionality of a Retrieval-Augmented Generation (RAG) model developed for multi-step CoP prediction in human gait analysis. Unlike typical sequence-to-sequence or LSTM models that rely solely on internal learning, this architecture leverages past data explicitly through a retrieval mechanism.

The RAG model is encapsulated in a modular class 'GaitRAGSystem' that performs five key stages: (1) data ingestion, (2) sequence embedding, (3) retrieval using FAISS, (4) prediction logic, and (5) visualization of predicted footprints.

1. Data Ingestion and Organization

The model begins by reading CoP gait data from a CSV file. The dataset contains joint coordinates and CoP (X, Y) values annotated by object ID and foot label (left/right). For each unique footstep, the CoP trajectory and joint positions are grouped and stored in dictionaries for easy access.

2. Embedding and Index Construction

To compare footprint sequences, each CoP sequence is embedded into a dense vector using the 'all-MiniLM-L6-v2' model from SentenceTransformers. Numerical CoP sequences are converted to string-like representations (e.g., "12.0,34.0 13.2,34.1 ...") before encoding. These embeddings are stored and indexed using FAISS (IndexFlatL2), enabling a fast nearest-neighbor search in vector space.

3. Retrieval of Similar Sequences

When a new CoP query sequence is presented (e.g., the last 10 known footsteps), the system computes its embedding and retrieves the top-k (here, $k = 5$) most similar se-

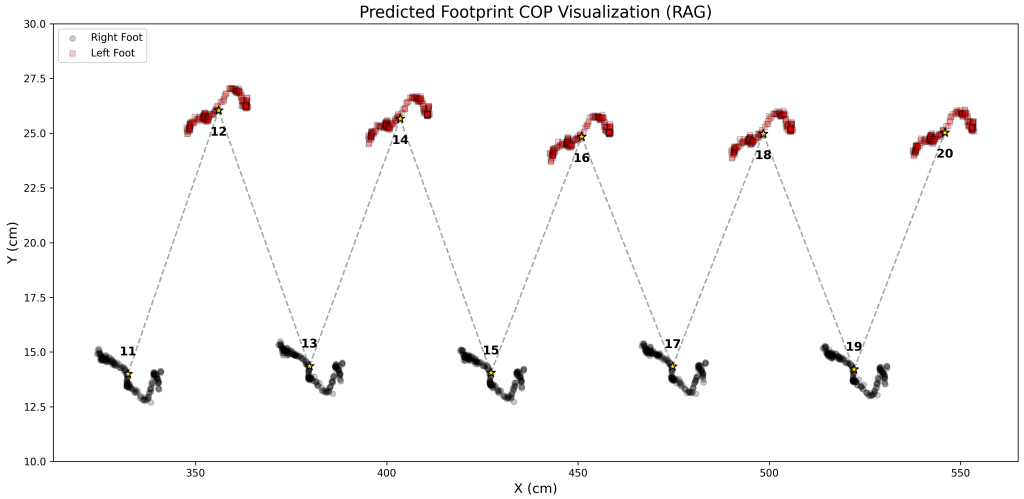

Figure 9.5: Visualization of predicted next 10 COP footprints using the RAG (Retrieval-Augmented Generation) model. The black and red dots represent the predicted foot pressure patterns for the right and left foot respectively. and the dashed lines show the estimated footstep sequence during gait.

quences from the FAISS index. Each retrieved sample contains a full CoP trajectory and joint data, which provide gait context.

4. Prediction Logic for Future Footsteps

The current RAG model predicts the next 10 footsteps using a stride-extrapolation logic rather than a neural generative decoder. It calculates the stride vector by analyzing the last two observed steps and uses this to iteratively project future steps. Each projected footprint position alternates between left and right foot, incorporating lateral swaying using a computed perpendicular vector. The shape of each footprint is generated by applying Gaussian noise to a normalized template footprint.

The model does not yet use a trained Transformer for CoP prediction, although a skeleton of such a model is implemented. Instead, it uses a mean-based synthetic prediction approach for realistic simulations.

5. Visualization

To evaluate prediction quality, the model plots each predicted footprint using plots (refer to Figure 9.5). Alternate steps are colored differently (black for right, red for left), and each footprint is annotated with its index in the sequence. This visual tool helps assess stride length, symmetry, and directional progression.

Figure 6.1 shows the actual CoP gait trajectory collected from real sensor data, while Figure 9.5 shows the predicted CoP positions for the next ten footsteps generated by the RAG model. Unlike LSTM or Seq2Seq models, the RAG system does not require input-output pairs for training. Instead, it works by retrieving similar past footstep patterns from a database and uses them to estimate what the next steps might look like.

Since we are predicting future footsteps and we do not have ground-truth data for those future steps, it is not possible to calculate standard evaluation metrics like RMSE or MAE. Even so, the results look reasonable. As seen in Figure 9.5, the predicted X-axis

values are higher than those in the actual data from Figure 6.1. This means the model is correctly predicts forward movement, which is the expected behavior.

However, there is a noticeable difference in the variation between the two figures. In the actual data (Figure 6.1), each step shows some natural variation in position and spacing. In contrast, the predicted steps from the RAG model appear very similar to each other, especially within left and right footsteps. This likely happens because the model was trained on a small amount of data and is using the average pattern of similar steps, rather than learning a full range of possible variations.

Overall, the RAG model is able to predict a forward walking pattern and maintain a left-right stepping rhythm. But due to the limited data, it tends to produce repeated patterns instead of realistic variations. In the future, having more diverse training data or using a more advanced generation method could improve this.

Summary

The RAG-based system enables data-efficient and interpretable gait forecasting by combining retrieval of semantically similar CoP sequences with stride-aware extrapolation. It provides next-10-footstep predictions and uses joint data contextually, making it well-suited for limited-data clinical applications. Future improvements may include training the Transformer-based prediction model using the retrieved data to further enhance temporal coherence and accuracy.

9.5 Gait Parameter Comparison Between GAITRite and RAG Predictions

To evaluate the biomechanical realism of the gait sequences generated by our Retrieval-Augmented Generation (RAG) model, we focused on computing essential spatial gait parameters: stride length, step length, and step width. These spatial descriptors offer insights into walking balance, symmetry, and stability. While both spatial and temporal metrics are typically considered in clinical gait analysis, our current implementation of the RAG model is not equipped to predict temporal dynamics such as step duration or cadence.

This limitation arises because the RAG model, in its current form, lacks a reliable mechanism to estimate future timestamps associated with Center-of-Pressure (CoP) trajectories. Temporal variations in human walking such as changes in step timing or acceleration are highly individualized and complex. Capturing such variations would require a significantly larger and more diverse dataset for the model to generalize well. Given this data constraint, our analysis has been limited to evaluating spatial parameters alone.

To conduct this evaluation, we extracted spatial gait parameters from the CoP coordinates predicted by the RAG model and compared them against the corresponding ground-truth parameters derived from the GAITRite system. GAITRite, a clinically validated pressure-sensitive walkway system, is widely regarded as a gold standard for gait measurement. The comparison enables us to assess how closely the synthetic CoP patterns generated by the RAG model approximate actual human walking behavior.

Based on the predicted CoP sequences, we computed the following average values for the spatial parameters:

1. **Average Left Stride Length:** 46.94 cm

2. **Average Right Stride Length:** 45.62 cm
3. **Average Step Length:** 23.73 cm
4. **Average Step Width:** 11.92 cm

For comparison, the GAITRite system provided the following average values:

1. **Average Left Stride Length:** 75.908 cm
2. **Average Right Stride Length:** 72.711 cm
3. **Average Step Length:** 37.4245 cm
4. **Average Step Width:** 12.33 cm

To quantify the deviation, we computed the percentage error using the formula:

$$\text{Percentage Error} = \left(\frac{|\text{Predicted} - \text{Actual}|}{\text{Actual}} \right) \times 100$$

The resulting percentage errors for each parameter are as follows:

Table 9.2: Percentage Error between RAG-Predicted and GAITRite-Measured Gait Parameters

Gait Parameter	**Percentage Error (%)**
Left Stride Length	38.14
Right Stride Length	37.26
Step Length	36.58
Step Width	3.32

These findings suggest that the RAG model, while effective at maintaining relative gait structure and forward progression, tends to underestimate absolute stride and step distances. This limitation can likely be attributed to the absence of physical calibration in the generative process and the limited volume of training data. Nonetheless, the step width predictions are closely aligned with the ground truth, reinforcing the model's ability to retain realistic lateral gait dynamics. Future improvements should incorporate gait constraints and camera calibration techniques to bridge the gap between synthetic and clinical measurements.

—

Chapter 10

Conclusions and Future Work

10.1 Conclusions

This thesis presented a vision-based machine learning pipeline for estimating the Center-of-Pressure (CoP) trajectory during human gait, using only RGB camera input and pose landmarks extracted via MediaPipe. The overall objective was to replicate GAITRite's clinical-grade measurements in a low-cost, non-invasive manner. Our key achievements and findings are summarized below:

- We developed a spatial registration pipeline based on Singular Value Decomposition (SVD) that successfully mapped image-space pose coordinates to real-world units with sub-centimeter accuracy.
- A synthetic gait data set was constructed using temporal interpolation from sparse footstep frames. This allowed for sufficient training data generation despite hardware constraints in continuous video acquisition.
- Multiple regression-based models including Multivariate Linear Regression, Random Forest, AdaBoost, and MLP - were evaluated as baselines. Among these, the Random Forest model performed best with an RMSE of approximately 1.72 cm.
- Deep sequential architectures such as LSTM and Seq2Seq were trained to capture temporal dependencies in pose sequences. The Seq2Seq model outperformed all others, achieving a consistent and low CoP prediction error (RMSE in the range of 0.38 - 0.50 cm), with smoother and temporally coherent trajectory outputs.
- Finally, we implemented a Retrieval Augmented Generation (RAG) framework to predict multiple future footsteps based on similarity search using FAISS and contextual averaging. While the RAG model did not use explicit sequence modeling, it generated spatially realistic and alternating footprints. The percentage error in the predicted gait parameters such as stride length and step width was reasonably low, with a step width error under 5%. This suggests the RAG model's utility in generative gait applications when trained on larger and more diverse datasets.

These contributions demonstrate the feasibility of a vision-only, machine-learning-based approach to gait analysis. If high feedality input data can be collected with further refinements such as real-time video input, better calibration, and scaling the proposed system could evolve into a practical and scalable alternative to traditional gait measurement technologies.

10.2 Future Work

We outline several avenues to extend and strengthen this research:

1. **Multi-View and Depth Fusion:** Integrate stereo or RGB-D cameras to reduce 2D occlusion errors and improve landmark position stability.
2. **Larger, Diverse Cohort:** Scale to 50+ subjects with varied gait pathologies to evaluate robustness and clinical relevance.
3. **Sensor Fusion:** Combine vision-based CoP estimates with wearable IMU data (accelerometers/gyros) for hybrid regression models.
4. **Real-Time Implementation:** Port the trained model to a mobile or embedded system for on-the-fly gait monitoring.

In conclusion, this work lays the groundwork for accessible, AI-driven gait analysis tools, democratizing Gait diagnostics without specialized hardware.

References

[1] Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Cham: Springer, 2018. ISBN: 9783319944623.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. ISBN: 9780387310732.

[3] Cognitive Creator. *Unveiling the Sigmoid Activation Function: A Thorough Exploration in Depth*. Accessed: 2025-05-26. 2023. URL: https://pub.aimind.so/unveiling-the-sigmoid-activation-function-a-thorough-exploration-in-depth-e84eba4b639f.

[4] Rahul Dey and Fathi M Salem. "Gate-variants of gated recurrent unit (GRU) neural networks". In: *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE. 2017, pp. 1597–1600.

[5] GAITRite. *GAITRite Electronic Walkway Technical Reference*. https://archive.org/details/manualzilla-id-5678389. Accessed: 2025-01-11. 2021.

[6] GO Classes. *GO Classes DA 2025 All India Mock Test 1, Question 31*. Accessed: 2025-05-26. 2025. URL: https://gateoverflow.in/448848/go-classes-da-2025-all-india-mock-test-1-question-31.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA: MIT Press, 2016. ISBN: 9780262035613.

[8] Klaus Greff et al. "LSTM: A search space odyssey". In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.

[9] Youssef Hosni. *Building RAG Application using Gemma 7B LLM & Upstash Vector Database*. Accessed: 2025-05-26. 2024. URL: https://youssefh.substack.com/p/building-rag-application-using-gemma.

[10] Thorir Mar Ingolfsson. *Insights into LSTM architecture*. Accessed: 2025-05-26. 2021. URL: https://thorirmar.com/post/insight_into_lstm/.

[11] Ranveer Joyseeree, Rami Abou Sabha, and Henning Mueller. "Applying machine learning to gait analysis data for disease identification". In: *Digital Healthcare Empowering Europeans*. IOS Press, 2015, pp. 850–854.

[12] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in neural information processing systems* 33 (2020), pp. 9459–9474.

[13] Yiqiao Lin, Xueyan Jiao, and Lei Zhao. "Detection of 3d human posture based on improved mediapipe". In: *Journal of Computer and Communications* 11.2 (2023), pp. 102–121.

[14] Jayanti Prasad. *Sequence to Sequence Model*. Accessed: 2025-05-26. 2023. URL: https://prasad-jayanti.medium.com/sequence-to-sequence-model-2c9d0fc4808.

[15] Rahm Ranjan et al. *Computer Vision for Clinical Gait Analysis: A Gait Abnormality Video Dataset*. 2024. arXiv: 2407.04190 [cs.CV]. URL: https://arxiv.org/abs/2407.04190.

[16] Marcelo Rudek et al. "A data-mining based method for the gait pattern analysis". In: *Facta Universitatis, Series: Mechanical Engineering* 13.3 (2015), pp. 205–215.

[17] Alex Sherstinsky. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306.

[18] Anirudha Sutar. *Deep Learning Explained: Neuronal Functions and Activation Functions*. Accessed: 2025-05-26. 2023. URL: https://medium.com/@sutaranirudha604/deep-learning-explained-neuronal-functions-and-activation-functions-82322c94c3d2.

[19] Bach Tran and Shivakumar Sastry. "Mapping a Virtual View to the Physical World to Guide the Completion of Complex Task Sequences". In: *2019 IEEE International Systems Conference (SysCon)*. 2019, pp. 1–8. DOI: 10.1109/SYSCON.2019.8836724.

[20] Lewis Tunstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, 2022. ISBN: 9781098103244.

[21] Author Unknown. *Multi-layer perceptron (MLP) NN basic Architecture*. https://www.researchgate.net/figure/Multi-layer-perceptron-MLP-NN-basic-Architecture_fig2_354817375. Accessed: 2025-05-26. 2021.

[22] Subhashini Venugopalan et al. "Sequence to sequence-video to text". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4534–4542.

[23] Timo Von Marcard et al. "Recovering accurate 3d human pose in the wild using imus and a moving camera". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 601–617.