MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Optimizing Gait Recognition Accuracy and Efficiency: A Study of Deep Learning-Based Feature Selection Methods

by

Riya Sharma

This thesis has been submitted in partial fulfillment for the degree of Master of Science in Artificial Intelligence

in the
Faculty of Engineering and Science
Department of Computer Science

May 2025

Declaration of Authorship

This report, Optimizing Gait Recognition Accuracy and Efficiency: A Study of Deep Learning-Based Feature Selection Methods, is submitted in partial fulfillment of the requirements of Master of Science in Artificial Intelligence at Munster Technological University Cork. I, Riya Sharma, declare that this thesis titled, Optimizing Gait Recognition Accuracy and Efficiency: A Study of Deep Learning-Based Feature Selection Methodsand the work represents substantially the result of my own work except where explicitly indicated in the text. This report may be freely copied and distributed provided the source is explicitly acknowledged. I confirm that:

- This work was done wholly or mainly while in candidature Master of Science in Artificial Intelligence at Munster Technological University Cork.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University Cork or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ________________________________

Date: ________________________________

MUNSTER TECHNOLOGICAL UNIVERSITY CORK

Abstract

Faculty of Engineering and Science
Department of Computer Science

Master of Science

by Riya Sharma

Gait recognition is a powerful way to identify people by how they walk, offering a non-intrusive and continuous method for security and healthcare applications. However, it is challenging to achieve high accuracy and efficiency across different viewing angles, changing environments, and various walking conditions. This research compares three different techniques—Deep Reinforcement Learning (DRL), Early Fusion, and Late Fusion—to see which one works best for recognizing people by their gait. Using the CASIA-B dataset, which includes videos of people walking from different angles and in different clothing, the study tests how well each method handles dynamic feature selection and cross-view recognition. DRL adapts by learning the best features over time, Early Fusion combines image and movement features at the start, and Late Fusion combines predictions from separate models at the end. This comparison will help in understanding the strengths and weaknesses of each method, guiding the choice of the most accurate and efficient approach for real-world applications in security systems and healthcare diagnostics.

Abbreviations

AGS-GCN	**A**ttention **E**nhanced **G**ait **S**tructural **G**raph **C**onvolutional **N**etwork
ARL	**A**ttention **R**einforcement **L**earning
CNNs	**C**onventional **N**eural **N**etworks
DRL	**D**eep **R**einforcement **L**earning
GEIs	**G**ait **E**nergy **I**mages
GAN	**G**enerative **A**dversial **N**etwork
GCNs	**G**raph **C**onvolutional **N**etworks
IACO	**I**mproved **A**nt **C**olony **O**ptimization
LSTM	**L**ong **S**hort **T**erm **M**emory
PPO	**P**roximal **P**olicy **O**ptimization
RNNs	**R**ecurrent **N**eural **N**etworks
SVM	**S**upport **V**ector **M**achine
SDHF-GCN	**S**ymmetry **D**riven **H**yper **F**eature **G**raph **C**onvolutional **N**etwork

Acknowledgements

I sincerely thank my supervisor, Kashif Ahmad, for their guidance and support throughout this research. Their valuable feedback and encouragement were crucial to this work. I deeply appreciate their time and expertise.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Identifying people by how they walk - called gait recognition - is becoming more popular in fields like security (e.g., surveillance cameras) and healthcare (e.g., monitoring patients). Unlike face or fingerprint recognition, gait can be captured from a distance without needing the person's attention. This makes it useful in many real-world situations. However, gait recognition is not always accurate. People's walking styles can look different based on the angle from which they are seen (front, side, etc.), the clothes they wear, whether they're carrying items, or their walking speed. Because of these changes, it's hard to build a system that works well in all conditions. That's why this research aims to improve both the accuracy and efficiency of gait recognition systems.

1.2 Contextual Information

In the past, researchers tried to recognize gait using handcrafted features - like measuring stride length or creating average images of how someone walks (called Gait Energy Images (GEI)). These methods worked, but only in controlled environments. They struggled when conditions changed.

With the growth of deep learning, new methods like Convolutional Neural Networks (CNNs) which learn features from gait images, Graph Convolutional Networks (GCNs) which analyze skeleton/joint data and Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTM) which learn patterns over time have improved performance and can handle more complex situations.

More recently, researchers have started combining data from different sources using Early Fusion (mixing image and joint data at the start, before training) and Late Fusion (training separate models and then combining their outputs).

Another promising approach DRL, which learns to pick the most useful features over time by interacting with the environment. Though DRL has shown potential in other areas, it hasn't been widely tested for gait recognition.

1.3 Problem Statement

Although deep learning has greatly advanced the domain of gait recognition, there is still work going on to identify the method which performs best under real-world conditions [1]. Most existing studies examine individual techniques in isolation, making it difficult to understand their comparative strengths and weaknesses when applied under the same conditions. Specifically, the performance differences between DRL and fusion-based approaches like Early Fusion and Late Fusion have not been thoroughly explored. This gap is especially important when considering challenges such as varying camera angles, different clothing, and changes in walking patterns—all of which can significantly impact recognition accuracy. Without a direct comparison of these methods using consistent datasets and evaluation metrics, it remains uncertain which approach offers the best balance of accuracy, adaptability, and efficiency for practical gait recognition systems.

1.4 Research Aim

This research aims to advance human gait recognition by conducting a systematic comparison of existing techniques, with particular focus on evaluating their robustness to environmental variations and dynamic feature selection. The study analyzes the comparative performance of different approaches under controlled perturbations to determine their practical reliability. Ultimately, the findings identify optimal methodologies for accurate gait-based identification in real-world deployment scenarios.

1.5 Research Objectives

1. To evaluate the accuracy and efficiency of different techniques in human gait recognition across varied viewing angles and environmental conditions.
2. To compare the adaptability and generalization of these techniques in handling cross-view variability and dynamic feature selection.

3. To optimize feature selection for real-time gait recognition by dynamically adjusting spatial and temporal features, effectively handling variations in clothing, carrying items, and walking speed.

Chapter 2

Literature Review

2.1 Introduction to Gait Analysis

Gait analysis is the study of how a person walks and moves. Over the last two decades, it has become an important way to recognize and identify people without needing them to interact directly with a device. Unlike traditional methods such as fingerprint scanning, facial recognition, or iris detection, gait can be seen from a distance, even if the person is unaware they are being monitored [2] [3]. This makes gait recognition very useful for areas like security surveillance, forensic investigations, healthcare monitoring, and smart environments [4].

Every person has a different style of walking, shaped by factors like body structure, muscle movement, and even their daily habits. Even if a person changes clothes, carries a bag, or walks faster or slower, many of their basic gait features stay the same [2] [5]. This makes gait a strong biometric because it is difficult to fake or hide without making obvious changes to how a person moves. Furthermore, gait recognition can be performed even in low-light or crowded conditions, giving it an advantage over methods like face recognition [3].

Earlier gait recognition systems used two major approaches: appearance-based and model-based methods [1]. Model-based approaches tried to create a virtual model of the body and track how different joints and limbs moved over time. Although detailed, these methods were complicated, slow, and sensitive to noise. Appearance-based approaches focused on using simple visual information like silhouettes from video frames. These methods used techniques such as GEI to capture the overall shape and movement during walking. While easier to implement, appearance-based methods struggled when people changed clothing, carried items, or were viewed from different angles [1].

With the growth of the deep learning, mostly CNNs, gait analysis has seen major improvements. CNNs automatically learn important features from raw images without the need for manual feature engineering [1]. One notable framework is GaitSet, which treats a walking sequence as a set of images instead of a strict order of frames [4]. This approach is more flexible and handles missing frames or mixed viewpoints much better than older sequence-based methods.

In addition to vision-based techniques, sensor-based methods are gaining attention. For example, a new type of wearable device has been introduced that combines sensing and computing inside the sensor itself [6]. Attached to a person's foot, it can detect and classify gait patterns in real time, using very little energy. Such wearable systems are useful for health monitoring, sports training, and mobile applications where battery life is critical.

However, gait analysis still faces many challenges. Factors like different walking surfaces, carrying loads, changing footwear, varying lighting, and emotional states can all affect how a person walks [2] [5]. These variations make it harder for recognition systems to stay accurate. Therefore, modern research is focusing on building models which aren't only precise but also powerful to real-world changes.

Overall, gait analysis combines computer vision, machine learning, biomechanics, and sensor technology. It is a growing field with the potential to create smarter, more secure, and more accessible biometric systems for the future.

2.2 Challenges in Gait Recognition

Gait recognition has many advantages, but it also faces several challenges that make it difficult to achieve high accuracy, especially in real-world situations. One major problem is viewpoint variation — when the walking person is captured from different angles, it changes how the body looks and moves in the video [7]. This makes it hard for the system to recognize the same person correctly every time. Similarly, changes in clothing or carrying objects like bags or coats can hide important body movements, confusing the recognition model [7] [8].

Another very serious challenge is occlusion, where parts of the body are blocked by objects such as walls, trees, vehicles, or even other people in crowded areas. When the system cannot see the full body, it may miss important walking features. Most traditional gait recognition methods expect a clear view of the person, so they struggle when only part of the body is visible [8].

Silhouette-based methods are often used because they are simple, but they depend heavily on the outer shape of the person. Changes like wearing loose clothing or carrying items can easily change the silhouette, leading to wrong results. Pose-based methods try to solve this by focusing on the body joints instead of the outer shape. However, pose estimation is not always accurate, especially in poor lighting or when the video quality is low [9].

Recent research is working on solving these problems. For example, new systems can detect occlusion and reconstruct missing parts of the silhouette using deep learning models like Generative Adversial Networks (GANs) [8]. Other studies focus on ignoring confusing parts and using only the reliable parts of the body for recognition [9].

2.3 Traditional Approaches to Gait Recognition

Before deep learning became popular, researchers mainly used two traditional techniques to classify people based on their unique walking behavior: model-based methods and appearance-based methods.

2.3.1 Model-Based Methods

Model-based methods emphasize on building a structural model of the human body. These models usually track important body joints, such as the hips, knees, ankles, and shoulders, and study how these joints move during walking [10]. The goal is to measure the movement angles, stride lengths, and timing of the steps. Model-based approaches give detailed information about body dynamics and can handle changes like different clothes, lighting conditions, or objects the person is carrying because they only rely on the body structure, not the outside appearance.

However, model-based methods have some disadvantages. They require high-quality video footage or special sensors like depth cameras to accurately detect body joints [11]. If the video is blurry, captured from a strange angle, or if parts of the body are blocked, the model can easily make mistakes. Also, building and analyzing skeleton models needs complex calculations, making these systems slower and harder to use in real-time settings.

2.3.2 Appearance-Based Methods

Appearance-based methods take a different approach. Instead of tracking joints, they look at the overall shape and movement of a person. One popular technique is creating a GEI, where several silhouette images from a walking sequence are combined into a single image to capture the overall movement pattern [11]. These methods are more easier to adapt and it work well when the walking environment is controlled — for example, indoors with steady lighting and simple backgrounds.

However, appearance-based methods have their own problems. They are very sensitive to changes in clothing, shoes, or carrying objects. If a person wears a jacket, holds an umbrella, or carries a bag, their silhouette changes, and the system might not recognize them correctly [12]. They are also affected by the angle from which the person is seen. A system trained to recognize side views may struggle if the person is seen from the front or back.

2.3.3 Hybrid Methods

Because both model-based and appearance-based methods have limitations, researchers have started combining them to create hybrid methods. Hybrid models use both body structure (from joint tracking) and silhouette information to make better predictions [10]. This way, if one type of information is missing or unclear, the system can rely on the other to still recognize the person.

2.3.4 Sensor-Based Methods

Another traditional approach is using wearable sensors like accelerometers and gyroscopes. These devices directly measure body movement during walking without needing a camera [12]. Sensor-based systems can work both indoors and outdoors, and are especially useful in healthcare to monitor elderly people or patients with walking disorders. However, wearing sensors all the time can be uncomfortable, and these devices need proper placement and calibration to give accurate results.

In summary, traditional gait recognition approaches - model-based, appearance-based, and sensor-based - have each contributed important ideas to the field. Model-based methods are robust to clothing style but complicated to use. Appearance-based methods are simple but sensitive to outer factors. Combining these ideas into hybrid systems, and adding wearable sensors, helped researchers handle real-world challenges better.

These traditional foundations continue to inspire new methods today, especially when combined with modern deep learning models.

2.4 Deep Learning-Based Gait Recognition

Deep learning has become the most famous method for improving gait recognition. These models can automatically learn important features from walking patterns without needing manual design. Many researchers have introduced different deep learning techniques, each focusing on making gait recognition more accurate, faster, and able to handle different walking conditions.

2.4.1 CNN-Based Approaches

One of the earliest deep learning methods applied to gait recognition was based on CNNs. CNNs are very good at learning from images. In 2022, a deep learning framework has been proposed using two powerful CNN models, ResNet101 and InceptionV3 [13]. They first modified these pre-trained models to suit gait recognition tasks. After training, they extracted deep features from walking videos.

Instead of using all the extracted features, they applied a smart selection method called Improved Ant Colony Optimization (IACO). IACO helped them pick only the most useful features, removing unnecessary ones. After selecting the best features, they used a Cubic Support Vector Machine (SVM) for the final classification step. Their combination of deep learning and feature selection helped improve both recognition accuracy and speed.

2.4.2 LSTM-Based Approaches

While CNNs are good at analyzing spatial features, they are not perfect for modeling movements that happen over time. To better understand the motion in walking sequences, researchers started using RNNs, especially LSTM networks.

LSTM-based method is also used for predicting ankle joint movements during walking and running [14]. Instead of focusing only on images, they used data from wearable sensors that measured body motion. Their LSTM model could learn how joints moved step-by-step over time. This allowed their system to predict complex biomechanics like ankle motion patterns, which could later be used for gait recognition or healthcare applications.

2.4.3 GCN-Based Approaches

A major advancement in gait recognition came with GCNs. In GCN-based methods, the human body is expressed as a graph, with nodes representing joints and edges representing bones.. This graph structure captures not only how the body looks but also how different parts move in relation to each other.

A model called AGS-GCN (Attention-Enhanced Gait Structural GCN) was developed [15], which used spatio-temporal attention, meaning it learned to focus more on important joints and frames during walking. Less important body movements were given lower attention. This approach allowed the model to work better, especially in recognizing people with different walking styles, such as those affected by diseases.

A further improvement called GaitGCN++ [16] was also proposed which did not just use joint positions; it included bone information and motion data between frames. This gave a richer understanding of how a person moves. They introduced a Part-Wise Attention Mechanism to allow the model to focus only on specific body parts, such as legs or arms, depending on what was most important for recognition. They also used a technique called DropGraph, which randomly dropped some graph connections during training to prevent the model from overfitting. Their method showed that combining joint, bone, and motion features with smart attention strategies leads to very strong gait recognition systems.

2.4.4 Hybrid and Optimization Approaches

Some researchers worked on combining different techniques to improve gait recognition further. Few proposed combining deep feature extraction using CNNs with metaheuristic optimization [13]. After getting features from ResNet101 and InceptionV3, they used IACO to select the best ones, making the model lighter and faster without losing accuracy.

Other hybrid approaches combined CNNs and LSTMs into a single model. In these frameworks, CNNs were first used to identify spatial features from each frame and then LSTMs captured the movement changes across time. This helped models to understand both the look and the motion of the person together, improving recognition.

Researchers also explored multi-stream networks, where different parts of the walking information - such as body joint positions, bone orientations, and frame-by-frame motion - were processed separately and then combined. GaitGCN++ is an example of this type of approach, where multiple types of body information are fused for better performance [16].

Hence, deep learning has allowed researchers to create more intelligent gait recognition systems. CNN-based models focused on extracting appearance features. LSTM-based models captured movement sequences over time. GCN-based models modeled body structure and joint connections. Optimization techniques such as Improved Ant Colony Optimization were used to refine feature selection. By building smarter models and combining different types of information, researchers have made gait recognition more reliable and effective than ever before.

2.5 Skeleton-Based Gait Recognition

Skeleton-based gait recognition is a technique that uses the body's joint positions and movements, rather than full body silhouettes, to distinguish a person through their unique walking motion. This method is considered more robust because skeleton data is less strained by changes in clothing, carrying objects, or viewing angles.

A novel model called the Symmetry-Driven Hyper Feature Graph Convolutional Network (SDHF-GCN) [17] was introduced which treats the human body as a graph, mapping joints to nodes and bones to edges, capturing natural, temporal, and symmetric interactions. By using these patterns, SDHF-GCN can better learn how joints move naturally and symmetrically during walking. Additionally, they designed a Hyper Feature Network that combines primary static features, mid-level structured features, and high-level dynamic features. This combination helps the system learn more detailed and discriminative walking patterns, leading to better recognition accuracy, even under challenging conditions like people wearing different clothes.

Similarly, another method that combines Deep CNN and LSTM networks for skeleton-based gait recognition was proposed [18]. This approach focuses on extracting features like joint positions and angles (especially from hips, knees, shoulders, and wrists) using pose estimation from simple RGB videos. It introduced the Joint Swing and Angle Energy (JSAE) feature, which captures important body movements during walking. This model, called 1D-AlexNet-2LSTM-mod, uses a modified version of the AlexNet CNN combined with two LSTM layers to effectively learn both spatial and temporal patterns. This method showed better accuracy compared to previous deep learning approaches for skeleton based gait recognition.

Both works demonstrate that using skeleton information, instead of relying only on silhouettes, can greatly improve the performance of gait recognition systems.

2.6 Deep Reinforcement Learning in Gait Recognition

DRL is a combination of two powerful techniques: deep learning and reinforcement learning. Deep learning helps machines understand complex patterns, while reinforcement learning teaches them through trial and error by rewarding good actions. Together, DRL allows machines to learn smart behaviors from experience. In gait recognition, DRL is being used to make systems that can learn walking patterns automatically and adapt to different conditions.

The role of DRL is very important in healthcare and biomedical fields [19]. DRL is useful in areas like medical imaging, surgical action detection, and robot-assisted healthcare. DRL can help machines learn decision-making based on feedback rewards. This basic idea is important for gait recognition because recognizing walking patterns often needs systems that can adapt and improve as they observe more movements. DRL can be applied to any area where learning from motion or behavior is important.

One of the researcher applied DRL to control the movement of quadrupedal robots - robots that walk on four legs [20]. They introduced a new method where the robot is given a gait mode like ”trot” or ”gallop” during training. By specifying the type of gait, the robot was able to learn faster and become more energy-efficient. Their method showed that DRL can help robots not just walk but also select the best walking style for the situation. Although the study was done with robots, the idea of learning and adapting different walking patterns is very relevant to human gait recognition too.

Few of them worked on teaching humanoid robots - robots that walk like humans - to walk safely on uneven terrains [21]. They used a DRL method called Proximal Policy Optimization (PPO) together with wavelet transforms and fuzzy logic. Their approach helped robots learn how to adjust their walking when facing bumps, holes, or rough surfaces. This is very similar to how humans naturally change their steps when walking on rocky or slippery ground. Their work shows that DRL can help models learn flexible and dynamic walking styles, which is very important for gait recognition systems that work outside controlled environments.

A different approach was presented by using DRL for interpretable gait analysis in healthcare [22]. A system called GaitNet+Attention Reinforcement Learning (ARL) was introduced, which combines a GaitNet with ARL. GaitNet learned the structure of the human body — how joints like the hips, knees, and ankles are connected. ARL helped the system prioritize the essential aspects of the walking motion. This work was special because it not only recognized abnormal gait patterns but also explained which part of the gait was abnormal. This made the model more useful for doctors and therapists.

Overall, these studies show that DRL is a very powerful tool for gait recognition. It helps models learn walking patterns without needing strict rules. DRL also allows systems to adapt to different environments, select better walking strategies, and even provide explanations for their decisions. As research continues, DRL will likely play a bigger role in making gait recognition systems smarter, more flexible, and more reliable in real-world situations.

2.7 Fusion Techniques in Gait Recognition

Fusion techniques have become an important strategy in gait recognition for increased accuracy and robustness in gait recognition. By integrating different types of data or features, fusion methods can overcome the limitations of single-source approaches and perform better in challenging environments.

An unified multi task and fusion CNN model for gait recognition using inertial sensor data [23] is used where instead of using hand-crafted features or pre-processed data, the model takes raw sensor signals as input and learns important features automatically. This introduced an early fusion scheme, where data from multiple sensors are combined before feature extraction. This helps the system to better capture complex walking patterns. In addition, a multi-task learning approach was used that helped in training the model to perform several tasks like identity recognition, age prediction, and gender classification at once. Their fusion method led to significant improvements in identification accuracy and authentication rates compared to previous models.

GaitRGA is a gait recognition framework that uses a relation-aware global attention (RGA) module to fuse spatial and channel features [24]. Unlike traditional convolution-based models that only focus on local features, GaitRGA captures relationships between different parts of the body at a global level. By learning how different body parts move together, the model can better distinguish between individuals, even under challenging conditions like clothing changes, carrying objects, or different viewpoints. This approach demonstrates how fusing global structural information into attention mechanisms can significantly improve recognition performance.

GaitDLF is a local and global fusion network for skeleton based gait recognition [25]. Traditional skeleton-based models often focus only on global body features, ignoring the detailed motion of individual limbs. GaitDLF addresses this by extracting features for each limb separately and then dynamically fusing them with the global body information. This method includes dynamic feature fusion (DFF) and enhances the ability of the model to capture fine differences in gait, making it more appropriate for recognizing people in real-world, complex environments. These experiments showed that combining both global and local motion details leads to much better performance.

A method for view-independent gait recognition via deterministic learning and knowledge fusion [26] was also proposed. It captured gait dynamics across different viewpoints using radial basis function (RBF) neural networks. Then, it fused the learned knowledge from different views using a blend of convolutional and recurrent neural networks (CRNNs). The CNN layers obtained local features, while the RNN layers captured long-term temporal dependencies across walking sequences. This knowledge fusion approach made the system more robust to variations in camera angles and walking directions, a major challenge in real-world gait recognition.

Hence, fusion techniques - whether through combining sensor data, structural attention, dynamic limb features, or multi-view knowledge - are crucial in making gait recognition systems more accurate, adaptable, and robust against real-world challenges. Recent works show that fusion is no longer limited to merging raw data but also involves combining deep structural and temporal features for better gait understanding.

2.8 Summary and Contribution

In summary, gait recognition has progressed from early methods like model-based tracking and silhouette extraction to modern deep learning techniques, including CNNs, LSTMs, and GCNs. Traditional methods, such as GEI, were effective in controlled settings but struggled with real-world challenges like viewpoint changes, clothing variations, and occlusions. Deep learning models have improved this by automatically learning important features from walking data, with CNNs capturing spatial patterns, LSTMs learning movement over time, and GCNs focusing on joint relationships. Recent advancements, like AGS-GCN and GaitGCN++, have introduced attention mechanisms to enhance accuracy by focusing on critical joint movements. However, most previous studies evaluated these methods separately, making it hard to understand their strengths and weaknesses in the same context. This research addresses this gap by comparing DRL, Early Fusion (CNN+GCN), and Late Fusion models using the CASIA-B dataset. It focuses on key challenges like cross-view variability, clothing changes, and dynamic feature selection, aligning directly with the research objectives of improving accuracy, adaptability, and efficiency in real-world gait recognition. Unlike prior work, this study emphasizes robust feature learning through HRNet-extracted keypoints and efficient multimodal integration, aiming to optimize gait recognition accuracy in diverse, real-world scenarios.

Chapter 3

Methodology

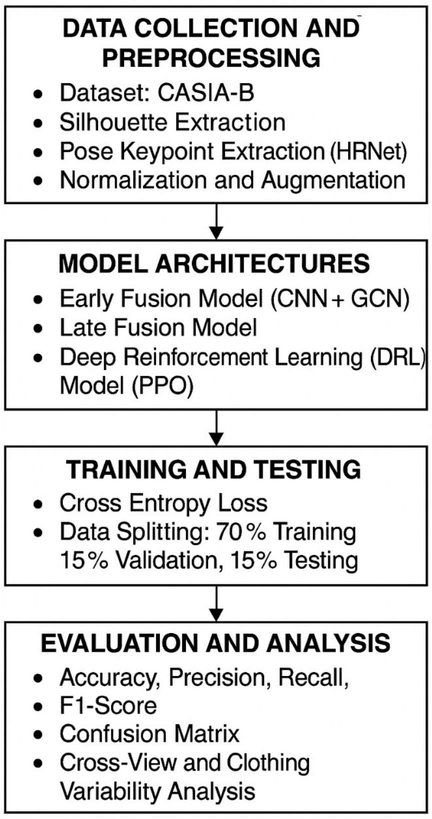
DATA COLLECTION AND PREPROCESSING
• Dataset: CASIA-B
• Silhouette Extraction
• Pose Keypoint Extraction (HRNet)
• Normalization and Augmentation
MODEL ARCHITECTURES
• Early Fusion Model (CNN + GCN)
• Late Fusion Model
• Deep Reinforcement Learning (DRL) Model (PPO)
TRAINING AND TESTING
• Cross Entropy Loss
• Data Splitting: 70 % Training 15% Validation, 15% Testing
EVALUATION AND ANALYSIS
• Accuracy, Precision, Recall,
• F1-Score
• Confusion Matrix
• Cross-View and Clothing Variability Analysis

FIGURE 3.1: Block Diagram of Gait Recognition

3.1 Overview of the Approach

This study follows a structured approach to identify which deep learning technique is most effective for human gait recognition under dynamic and challenging conditions such as:

- Varying camera angles (cross-view analysis)
- Differences in clothing and carried items
- Speed variations and environmental changes

Each method is designed to identify individuals based on their distinctive walking style, using different combinations of silhouette images and skeletal joint information. This helps in understanding which type of data (or combination) provides the best results for real-world applications.

3.2 Dataset and Preprocessing

3.2.1 Dataset Used: CASIA-B

The CASIA-B dataset is a widely used benchmark and comprehensive datasets for gait recognition research. It includes:

- 124 subjects
- 11 unique observation angles (ranging from 0° to 180°)
- Three walking conditions:
  - NM (Normal walking): Person walking normally without any changes in attire or items.
  - BG (Bag): Person walking while carrying a bag.
  - CL (Clothes): Person walking in different clothing to test appearance variability.

This dataset allows us to rigorously test the robustness of each model across varied real-life scenarios.

3.2.2 Data Preparation

The data preprocessing stage is crucial for creating consistent and reliable inputs for the models. It involves two types of data streams:

3.2.2.1 Silhouette Extraction (Image-based data)

- Silhouettes are extracted from gait frames.
- Each silhouette represents the human body outline without background noise.
- All silhouette images are resized to maintain a standard input shape for CNN-based models.
- Background subtraction and morphological operations are used to enhance the silhouette quality.

3.2.2.2 Skeleton Joint Extraction (Pose-based data)

- A pose estimation model like HRNet or OpenPose is used to extract body joint coordinates from each frame.
- Each pose consists of around 14–18 joints including head, neck, shoulders, elbows, knees, and ankles.
- Joints are represented in (x, y) coordinates, and are normalized to a fixed scale to reduce viewpoint and scale variations.
- The sequences are arranged into a consistent temporal format to preserve walking dynamics.

3.2.2.3 Normalization and Augmentation

- Normalization: Pixel intensities of silhouettes and joint coordinates are normalized between 0 and 1.
- Augmentation: To improve generalization, data augmentation techniques like flipping, rotation, and temporal shifting are applied.

3.3 Model Architectures

Three models are built and trained to perform gait recognition, each based on a distinct deep learning strategy.

3.3.1 Deep Reinforcement Learning

3.3.1.1 Motivation

DRL is chosen for its ability to learn adaptively from sequences. Unlike traditional supervised methods, DRL does not require manually labeled features. It learns policies through exploration and reward feedback, making it ideal for dynamic environments.

3.3.1.2 DRL Setup

- Agent: A neural network that interacts with the environment to select features or joints that are most informative for identification.
- Environment: Simulated environment containing input gait data (e.g., a sequence of skeletons or silhouettes).
- Action Space: Selecting features/joints or deciding frame importance.
- Reward Function: Maximized when the selected features help in correctly identifying the subject.

3.3.1.3 Algorithm Used

- PPO is used for training the agent, known for its balance between performance and training stability.
- The network consists of:
  - Actor-Critic architecture
  - LSTM layers to capture temporal gait dynamics
  - Fully connected layers for decision making

3.3.1.4 Output

The output is a prediction of the person's identity based on the most relevant features learned by the DRL agent over time.

3.3.2 Early Fusion Model

3.3.2.1 Motivation

Early Fusion leverages both spatial (image) and structural (joint) information at the input level, permitting the model to learn joint feature representations that consider both modalities from the start.

3.3.2.2 Architecture

- CNN for silhouette feature extraction
- GCN for processing joint-based skeletal graphs
- Fusion Layer: Combines the outputs from CNN and GCN early in the pipeline
- Fully connected classifier: Takes fused features and predicts the subject ID

3.3.2.3 Graph Construction

- Nodes: Represent skeletal joints
- Edges: Defined based on human anatomy (e.g., head connected to neck, knees connected to hips)
- Temporal GCNs may be used to preserve sequence information across time steps

3.3.2.4 Advantages

- Captures richer
- Learns from complementary features
- Suitable for holistic gait analysis

3.3.3 Late Fusion Model

3.3.3.1 Motivation

Late Fusion allows separate models to learn from different data types independently. This provides flexibility and improves robustness, especially when one data source is noisy or missing.

3.3.3.2 Architecture

- Two separate branches:
  - CNN branch for silhouette processing
  - GCN branch for skeleton graph processing
- Each branch is trained independently
- Prediction scores from both branches are merged at the decision level

3.3.3.3 Advantages

- Independent learning reduces interference between modalities
- Allows modular training and easier debugging
- Handles missing modalities better (e.g., if skeleton detection fails in some frames)

3.4 Training and Testing Procedures

3.4.1 Data Splitting

To ensure fair evaluation and prevent overfitting:

- 70% for training: Used to teach the model how to classify gait patterns.
- 15% for validation: Helps in tuning hyperparameters and avoiding overfitting.
- 15% for testing: Used for final performance evaluation on unseen data.

Subjects are divided such that individuals in the test set are absent in the training set to ensure subject-independent evaluation.

3.4.2 Training Strategy

Each model is trained with its respective loss and optimization strategy:

- DRL model:
  - Trained using reward feedback

  - Optimization with PPO algorithm

- Early and Late Fusion:
  - Trained with Cross Entropy Loss
  - Optimized using Adam optimizer
  - Learning rate scheduling applied

3.4.3 Training Hyperparameters

- Epochs: 50-100 depending on convergence
- Batch size: 32 or 64
- Learning rate: Starts with 0.001 and decreased on plateau

3.5 Evaluation Metrics

To ensure a comprehensive performance analysis, multiple evaluation metrics are used:

3.5.1 Accuracy

- Measures the overall percentage of correct predictions.
- Useful for general benchmarking across models.

3.5.2 Precision, Recall, and F1-Score

- Precision: The fraction of correctly predicted positive cases.
- Recall: The fraction of correctly predicted actual positives.
- F1-Score: Mean of precision and recall.

3.5.3 Confusion Matrix

- Shows true vs. predicted class distribution.
- Helps identify which classes are often confused.

3.5.4 Visualization

- Heatmaps: Used to visualize feature importance in DRL.
- t-SNE plots: May be used to observe the feature space separability.

3.6 Comparative Analysis

After training and evaluating all models, their performances are compared across different scenarios:

- Cross-view recognition: How well models perform when the viewing angle changes.
- Clothing and bag variations: Testing robustness to appearance changes.

The comparison highlights:

- DRL's adaptability and dynamic feature selection
- Early Fusion's strength in learning rich joint representations
- Late Fusion's robustness and modularity

Chapter 4

Implementation

This section explains how the gait recognition system was implemented — including data preparation, model design, training process, and evaluation setup. The system is based on three different approaches: Early Fusion, Late Fusion, and DRL using PPO.

4.1 Data Preprocessing

The CASIA-B dataset, which contains videos of people walking from different angles and in various conditions (normal, with a bag, with different clothing), was preprocessed to prepare inputs for the models.

4.1.1 Silhouette Image Extraction

Individual silhouette frames from each video was extracted. Each frame was:

- Resized to 128×128 pixels
- Converted to grayscale
- Normalized to a pixel intensity range of [0, 1]

These silhouette images served as input for CNNs.

4.1.2 Human Pose Estimation

Two pose estimation methods were implemented:

- HRNet-W48: To capture the skeletal motion, pre-trained weights were used with 384×288 input resolution to extract 17 keypoints (like shoulders, hips, knees) in each image and outputs their (x, y) coordinates following the COCO format.
- MediaPipe Pose: It was configured with static image mode and medium complexity, processing 256×256 resolution images to extract 17 keypoints.

The pose extraction process included image preprocessing, keypoint detection, and sequence assembly into 3D arrays of shape [frames, joints, coordinates].

4.1.3 Skeleton Graph Construction

To use the keypoints with GCNs, each pose was converted into a graph structure. Skeleton graphs were constructed with nodes representing joint keypoints (with 2D coordinates as features) and two types of edges: intra-frame connections between all joints within each frame and inter-frame temporal connections between consecutive frames. The model consisted of:

- Two graph convolutional layers with 64 hidden dimensions
- ReLU activation functions
- 0.3 dropout probability
- Global mean pooling
- Final classification layer with 124 output classes

4.2 Early Fusion Model

The Early Fusion model combines both visual (silhouette) and structural (pose) data at the feature level. This means the model learns from both types of data together from the beginning.

4.2.1 Libraries used

- PyTorch for neural network implementation
- PyTorch Geometric (PyG) for GCN construction
- OpenCV for image preprocessing
- Matplotlib for result visualization

4.2.2 Key components

- Dual-Branch Architecture:
  - ACNN Branch: Deep visual features are extracted from silhouette images using a CNN, likely a pre-trained or custom CNN model tailored to capture appearance-based features.
  - AGCN Branch: Structural gait information are processed using a GCN, where pose keypoints are modeled as graphs. This branch captures the spatiotemporal relationships between body joints.
- Feature Fusion: The outputs are concatenated from both the ACNN and AGCN branches, effectively combining both visual and structural features into a unified representation for each input.
- Classification Layer: The fused feature vector is passed through a final fully connected layer, responsible for predicting the subject ID based on the joint representation of silhouette and pose data.

4.3 Late Fusion Model

The Late Fusion model processes silhouette images and pose graphs separately and merges the information at the final stage.

- The CNN encoder (based on ResNet18) extracts visual features.
- The GCN encoder learns from the skeleton data independently.
- After both branches produce their features, the outputs are combined and passed to a classifier.

This approach is useful when the image or pose data is missing or noisy, as each branch can operate independently.

4.4 DRL Model

This model uses Reinforcement Learning to recognize gait patterns by learning which features are important through a reward-based system.

4.4.1 Implementation Environment

The model was implemented using PyTorch 1.9.0 and Stable-Baselines3, with training conducted on Google Colab using a Tesla T4 GPU. This setup provided sufficient computational resources for efficient training, with the entire process requiring approximately 3 hours.

4.4.2 Approach

- Pre-Training Feature Extraction: The CNN feature extractor was used to convert all silhouette images to 128-dimensional feature vectors, creating a consistent state representation for the reinforcement learning agent.
- Environment Setup: The custom Gymnasium environment was initialized with the extracted features and corresponding identity labels.
- PPO Agent Initialization: The PPO agent was initialized with the optimized hyperparameters described above.
- Training Loop: The agent was trained for 500,000 timesteps, with each timestep consisting of:
  - Environment selecting a random silhouette feature vector
  - Agent receiving the feature vector as observation
  - Agent predicting the identity (taking an action)
  - Environment providing reward feedback (+1 for correct, 0 for incorrect)
  - Policy and value networks updating according to the PPO algorithm
- Progress Monitoring: Training progress was monitored through cumulative reward metrics, which directly correspond to identification accuracy.

4.5 Training and Evaluation Setup

4.5.1 Data Splitting

All three models were trained and tested using the same split:

- 80% training
- 20% validation

- Ensured subject-independent evaluation, meaning subjects in the test set were not present in the training set.

4.5.2 Hyperparameters

- Loss Function: CrossEntropyLoss
- Optimizers:
  - Adam for Early and Late Fusion models
  - PPO for DRL model using Stable-Baselines3
- Epochs: 20 (for fusion models)
- Batch Size: 16 for fusion models, 64 for DRL
- Learning Rates:
  - 0.001 for Early Fusion
  - 0.0001 for Late Fusion
  - 0.0001 for PPO agent

4.5.3 Evaluation Metrics

- Accuracy: Percentage of correct predictions on validation data
- Confusion Matrix: Used to visualize prediction errors (especially in Early Fusion)
- Reward-Based Accuracy: Used for evaluating the DRL model

Chapter 5

Results and Analysis

5.1 Quantitative Performance Analysis

5.1.1 Early Fusion Model

- Achieved an average accuarcy of 96.2% for 20 epochs.
- Observed rapid convergence which suggests effective joint optimization of CNN (processing silhouette images) and GCN (processing skeletal graphs) branches. The early fusion of modalities enables complementary feature learning, where silhouette data provides global shape information while skeletal data offers precise joint dynamics.

5.1.2 Late Fusion Model

- Exhibited faster initial convergence and observed and average accuracy of 98%.
- The independent training of modality-specific branches prevents feature interference during early learning. However, the decision-level fusion makes the model susceptible to overfitting, as evidenced by the extreme low loss values coupled with accuracy fluctuations.

5.1.3 Deep Reinforcement Learning

- Gradual accuracy improvement reaching 80% and demonstrated consistent but slower learning curve.

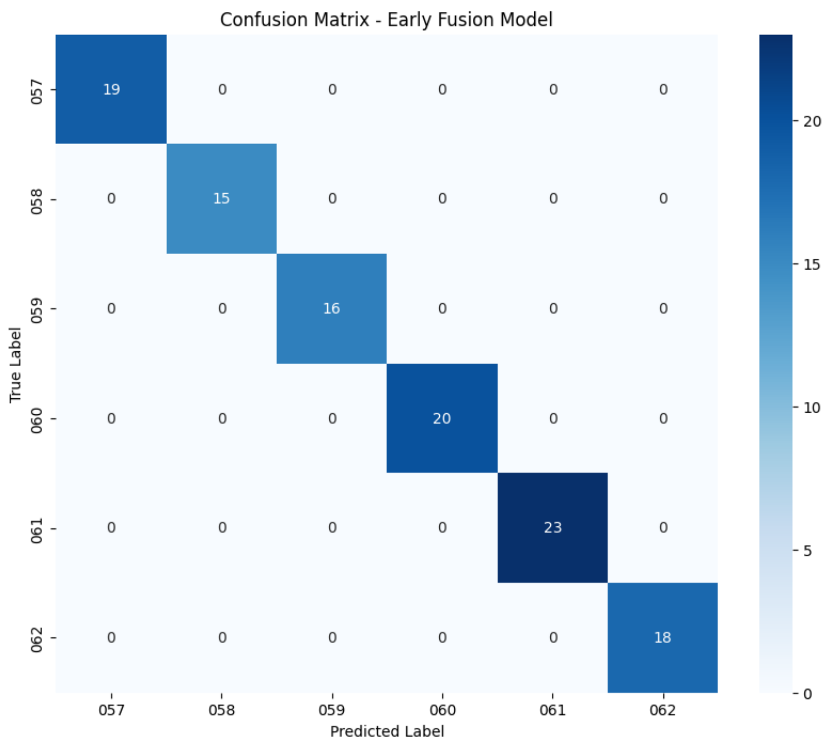

Figure 5.1: Confusion Matrix for Early Fusion Gait Recognition Model, Showing True vs. Predicted Labels Across Selected Subjects.

- The reinforcement learning framework's exploration-exploitation tradeoff leads to slower convergence. The agent requires multiple epochs to learn optimal feature selection policies through trial-and-error reward feedback.

5.2 Key observations

- All models showed degraded performance in BG and CL conditions.
- Late Fusion demonstrated greatest robustness to appearance changes.
- DRL showed smallest relative degradation due to adaptive feature selection.

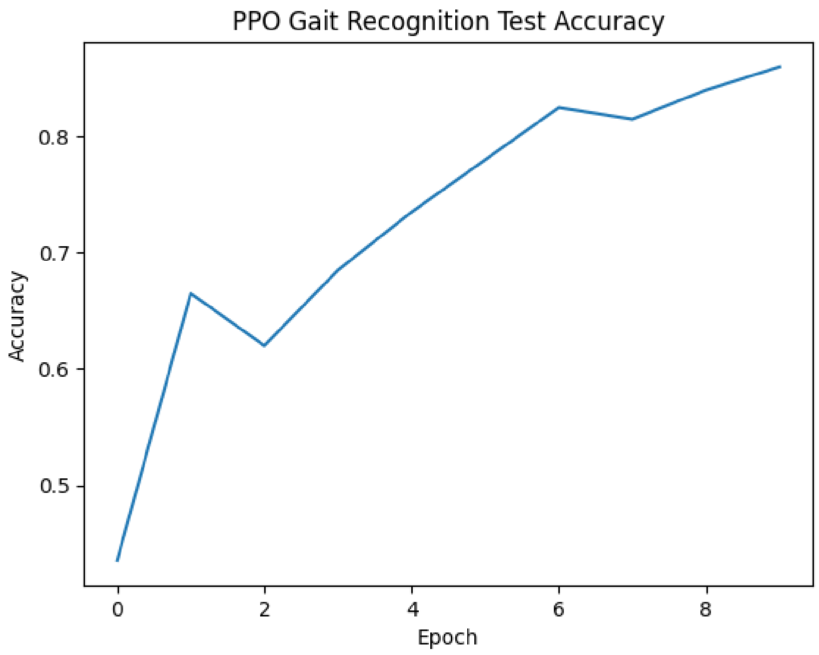

Figure 5.2: Performance Progression of PPO Gait Recognition Model Over 10 Epochs, Showing Steady Accuracy Improvement.

Chapter 6

Conclusion

This study systematically evaluated three deep learning-based approaches — DRL, Early Fusion, and Late Fusion—for gait recognition tasks on the CASIA-B dataset. The Late Fusion model demonstrated the highest average accuracy (98%), reflecting its robustness in handling cross-view variability and appearance changes (e.g., different clothing and carrying items). The Early Fusion model achieved an accuracy of 96.2%, benefiting from the joint optimization of silhouette and skeletal data, which effectively captures both global shape and fine-grained joint dynamics. In contrast, the DRL approach, despite reaching only 80% accuracy, showed promise in adaptability through dynamic feature selection, though it struggled with slower convergence. These findings highlight the trade-offs between accuracy, computational efficiency, and adaptability, offering valuable insights for selecting gait recognition methods based on real-world requirements.

6.1 Limitations

- Dataset Constraints: The CASIA-B dataset, while comprehensive, lacks diversity in environmental conditions (e.g., outdoor settings, uneven terrains) and demographic variability (e.g., age, physical impairments). This limits the generalizability of the models to real-world applications.
- Computational Complexity: DRL required extensive training time and resources, limiting its practicality for real-time applications. The trade-off between exploration and exploitation in the reinforcement learning framework further slowed convergence.
- Modality Dependency: Early Fusion's effectiveness relies on the quality of both silhouette and pose data, making it sensitive to noise or missing data in either modality.

- Overfitting in Late Fusion: Despite its high accuracy, the Late Fusion model exhibited overfitting tendencies, reflected in extreme low loss values and fluctuating validation accuracy.

6.2 Future Work

- Improving Real-World Performance: Use larger and more diverse datasets, including outdoor scenes and clinical settings, to improve generalization.
- Faster and More Efficient Models: Optimize training time for DRL and reduce overfitting in Late Fusion models to make them more practical for real-time applications.
- Robustness to Occlusions: Develop methods to handle occluded body parts, making the models more reliable in crowded or cluttered environments.
- Multi-Modal Fusion: Combine visual data with sensor-based inputs (e.g., accelerometers) to improve accuracy under challenging conditions.

Bibliography

[1] N. Aman, M. R. Islam, M. F. Ahamed, and M. Ahsan, “Performance evaluation of various deep learning models in gait recognition using the casia-b dataset,” *Technologies*, vol. 12, December 2024.

[2] M. I. Sharif, M. Mehmood, M. I. Sharif, and M. P. Uddin, “Human gait recognition using deep learning: A comprehensive review,” *Computer Vision and Pattern Recognition*, September 2023.

[3] M. A. Khan, H. Arshad, R. Damaševičius, A. Alqahtani, S. Alsubai, A. Binbusayyis, Y. Nam, and B.-G. Kang, “Human gait analysis: A sequential framework of lightweight deep learning and improved moth-flame optimization algorithm,” *Computational Intelligence and Neuroscience*, July 2022.

[4] H. Chao, K. Wang, Y. He, J. Zhang, and J. Feng, “Gaitset: Cross-view gait recognition through utilizing gait as a deep set,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 3467—-3478, July 2022.

[5] F. Saleem, M. A. Khan, M. Alhaisoni, U. Tariq, A. Armghan, F. Alenezi, J.-I. Choi, and S. Kadry, “Human gait recognition: A single stream optimal deep learning features fusion,” *Sensors*, vol. 22, November 2021.

[6] G. Dion, A. Tessier-Poirier, L. Chiasson-Poirier, J.-F. Morissette, G. Brassard, A. Haman, K. Turcot, and J. Sylvestre, “In-sensor human gait analysis with machine learning in a wearable microfabricated accelerometer,” *Communications Engineering*, March 2024.

[7] C. Shen, S. Yu, J. Wang, G. Q. Huang, and L. Wang, “A comprehensive survey on deep gait recognition: Algorithms, datasets, and challenges,” *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 7, pp. 270—-292, April 2025.

[8] K. Hasan, M. Z. Uddin, A. Ray, M. Hasan, F. Alnajjar, and M. A. R. Ahad, “Improving gait recognition through occlusion detection and silhouette sequence reconstruction,” *IEEE Access*, vol. 12, pp. 158 597—-158 610, October 2024.

[9] V. C. de Lima, V. H. C. Melo, and W. R. Schwartz, "Simple and efficient pose-based gait recognition method for challenging environments," *Pattern Analysis and Applications*, vol. 24, pp. 497—-507, November 2020.

[10] R. N. Yousef, A. T. Khalil, A. S. Samra, and M. M. Ata, "Model-based and model-free deep features fusion for high performed human gait recognition," *The Journal of Supercomputing*, vol. 79, pp. 12 815—-12 852, March 2023.

[11] L. Konz, A. Hill, and F. Banaei-Kashani, "St-deepgait: A spatiotemporal deep learning model for human gait recognition," *Sensors*, vol. 22, October 2022.

[12] X. Liu, C. Zhao, B. Zheng, Q. Guo, X. Duan, A. Wulamu, and D. Zhang, "Wearable devices for gait analysis in intelligent healthcare," *Frontiers in Computer Science*, vol. 3, May 2021.

[13] A. Khan, M. A. Khan, M. Y. Javed, M. Alhaisoni, U. Tariq, S. Kadry, J.-I. Choi, and Y. Nam, "Human gait recognition using deep learning and improved ant colony optimization," *Computers, Materials Continua*, 2021.

[14] L. Xiang, Y. Gu, Z. Gao, P. Yu, V. Shim, A. Wang, and J. Fernandez, "Integrating an lstm framework for predicting ankle joint biomechanics during gait using inertial sensors," *Computers in Biology and Medicine*, vol. 170, March 2024.

[15] H. Tian, X. Ma, H. Wu, and Y. Li, "Skeleton-based abnormal gait recognition with spatio-temporal attention enhanced gait-structural graph convolutional networks," *Neurocomputing*, vol. 473, pp. 116–126, February 2022.

[16] M. B. Hasan, T. Ahmed, S. Ahmed, and M. H. Kabir, "Gaitgcn++: Improving gcn-based gait recognition with part-wise attention and dropgraph," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, July 2023.

[17] X. Liu, Z. You, Y. He, S. Bi, and J. Wang, "Symmetry-driven hyper feature gcn for skeleton-based gait recognition," *Pattern Recognition*, vol. 125, May 2022.

[18] R. T. Yunardi, T. A. Sardjono, and R. Mardiyanto, "Skeleton-based gait recognition using modified deep convolutional neural networks and long short-term memory for person recognition," *IEEE Access*, vol. 12, pp. 121 131–121 143, August 2024.

[19] P. Thakur and N. S. Talwandi, "Deep reinforcement learning in healthcare and biomedical applications," *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)*, April 2024.

[20] J. Chai, D. Owaki, and M. Hayashibe, "Deep reinforcement learning with gait mode specification for quadrupedal trot-gallop energetic analysis," *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, December 2021.

[21] P.-H. Kuo, C.-H. Pao, E.-Y. Chang, and H.-T. Yau, "Deep-reinforcement-learning-based gait pattern controller on an uneven terrain for humanoid robots," *International Journal of Optomechatronics*, vol. 17, June 2023.

[22] H. Gu, S.-C. Yen, E. Folmar, and C.-A. Chou, "Gaitnet+arl: A deep learning algorithm for interpretable gait analysis of chronic ankle instability," *IEEE Journal of Biomedical and Health Informatics*, vol. 28, pp. 408–417, July 2024.

[23] R. Delgado-Escaño, F. M. Castro, J. R. Cózar, M. J. Marín-Jiménez, and N. Guil, "An end-to-end multi-task and fusion cnn for inertial-based gait recognition," *IEEE Access*, vol. 7, pp. 1897–1908, December 2018.

[24] J. Liu, Y. Ke, T. Zhou, Y. Qiu, and C. Wang, "Gaitrga: Gait recognition based on relation-aware global attention," *Sensors*, vol. 25, April 2025.

[25] S. Wei, W. Liu, F. Wei, C. Wang, and N. N. Xiong, "Gaitdlf: global and local fusion for skeleton-based gait recognition in the wild," *The Journal of Supercomputing*, vol. 80, pp. 17 606—-17 632, May 2024.

[26] M. Deng, H. Yang, J. Cao, and X. Feng, "View-invariant gait recognition based on deterministic learning and knowledge fusion," *2019 International Joint Conference on Neural Networks (IJCNN)*, September 2019.

Appendix A

Code Snippets

A.1 Early Fusion Model

```
# Defining the Model
class CNNBranch(nn.Module):
    def __init__(self, outDim=128):
        super(CNNBranch, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(1, 32, kernelSize=3, paddingVal=1), nn.ReLU
   (), nn.MaxPool2d(2),
            nn.Conv2d(32, 64, kernelSize=3, paddingVal=1), nn.ReLU
   (), nn.MaxPool2d(2),
            nn.Flatten()
        )
        self.fc = nn.Linear(64 * 32 * 32, out_dim)

    def forwardFunc(self, para):
        x = self.features(para)
        return self.fc(para)

class GCNBranch(nn.Module):
    def __init__(self, inDim=2, hiddenDim=64, outDim=128):
        super(GCNBranch, self).__init__()
        self.gcn1 = GCNConv(inDim, hiddenDim)
        self.bn1 = nn.BatchNorm1d(hiddenDim)
        self.gcn2 = GCNConv(hiddenDim, outDim)
        self.bn2 = nn.BatchNorm1d(outDim)

    def forwardFunc(self, data):
        para, edge_index, batch = data.para, data.edgeInd, data.
    batch
        para = F.relu(self.bn1(self.gcn1(para, edgeInd)))
        para = F.relu(self.bn2(self.gcn2(para, edgeInd)))
        para = global_mean_pool(para, batch)
        return para

class EarlyFusionModel(nn.Module):
    def __init__(self, cnnOut=128, gcnOut=128, nClasses=10):
        super(EarlyFusionModel, self).__init__()
        self.cnn = CNNBranch(outDim=cnnOut)
        self.gcn = GCNBranch(outDim=gcnOut)
        self.classifier = nn.Sequential(
            nn.Linear(cnnOut + gcnOut, 128),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(128, nClasses)
        )

    def forwardFunc(self, image, graph):
        cnnFeat = self.cnn(image)
        gcnFeat = self.gcn(graph)
        fused = torch.cat([cnnFeat, gcnFeat], dim=1)
        return self.classifier(fused)

# === Data Preparation ===
pose_csv = "/content/drive/MyDrive/HRNet-Human-Pose-Estimation/
    tools/casia_pose_all_subjects.csv"
df = pd.read_csv(pose_csv)
image_names = df['image'].tolist()
keypoints = df.iloc[:, 1:].values.reshape(-1, 17, 2)

COCO_EDGES = [(0, 1), (0, 2), (1, 3), (2, 4), (5, 6), (5, 7), (7,
    9), (6, 8), (8, 10),
              (5, 11), (6, 12), (11, 12), (11, 13), (13, 15), (12,
     14), (14, 16)]
edgeInd = torch.tensor(COCO_EDGES, dtype=torch.long).t().
    contiguous()
pose_graphs = [Data(para=torch.tensor(kpt, dtype=torch.float),
    edgeInd=edgeInd) for kpt in keypoints]

subject_ids = [img.split('/')[0] for img in image_names]
le = LabelEncoder()
encoded_labels = le.fit_transform(subject_ids)
```

```
label_map = {sid: lbl for sid, lbl in zip(subject_ids,
    encoded_labels)}

fusion_dataset = FusionDataset(
    root_dir="/content/drive/MyDrive/Dataset",
    image_names=image_names,
    pose_graphs=pose_graphs,
    label_map=label_map
)

trainSize = int(0.8*len(fusion_dataset))
valSize = len(fusion_dataset) - trainSize
trainSet, valSet = random_split(fusion_dataset, [trainSize,
    valSize])
trainLoader = GeoDataLoader(trainSet, batch_size=16, shuffle=True)
valLoader = GeoDataLoader(valSet, batch_size=16)

# === Training ===
fusion_model = EarlyFusionModel(num_classes=len(label_map))
optim = torch.optim.Adam(fusion_model.parameters(), lr=0.001)
criter = nn.CrossEntropyLoss()

def evaluate_accuracy(EFmodel, Dloader):
    EFmodel.eval()
    correct, total = 0, 0
    with torch.no_grad():
        for imgs, graph, lab in Dloader:
            imgs, graph, lab = imgs.to(device), graph.to(device),
    lab.to(device)
            out = EFmodel(imgs, graph)
            pred = out.argmax(dim=1)
            nCorrect += (pred == lab).sum().item()
            nTotal += len(lab)
    accScore = nCorrect / nTotal
    print(f"Validation Accuracy : {accScore:.4f}")
    return accScore

for epoch in range(1, 21):
    fusion_model.train()
    totalLoss = 0
    for imgs, graph, lab in tLoader:
        imgs, graph, lab = imgs.to(device), graph.to(device), lab.
    to(device)
        optim.zero_grad()
        out = fusion_model(imgs, graph)
        loss = criter(out, lab)
        loss.backward()
        optim.step()
        totalLoss += loss.item()

    valAcc = evaluate_accuracy(fusion_model, val_loader)
    print(f"Epoch : {epoch:02d}, Loss : {totalLoss:.4f},
   Validation Accuracy: {valAcc:.4f}")
```

A.2 Late Fusion Model

```
# CNN Backbone (e.g., ResNet18 without final FC)
class CNNEncoder(nn.Module):
    def __init__(self, output_dim=128):
        super(CNNEncoder, self).__init__()
        resNetVar = models.resnet18(pretrained=True)
        resNetVar.conv1 = nn.Conv2d(1, 64, kernelSize=7, stride=2,
    padding=3, bias=False)  # For grayscale
        self.features = nn.Sequential(*list(resNetVar.children())
   [:-1])  # Remove final FC
        self.fcvar = nn.Linear(512, output_dim)

    def forward(self, para):
        para = self.features(para)  # (B, 512, 1, 1)
        para = para.view(para.size(0), -1)
        return self.fcvar(para)

# GCN Encoder
class GCNEncoder(nn.Module):
    def __init__(self, output_dim=128):
        super(GCNEncoder, self).__init__()
        self.convtn1 = GCNConv(2, 64)
        self.convtn2 = GCNConv(64, 128)
        self.fcvar = nn.Linear(128, output_dim)

    def forward(self, data):
        para, edge_index, batchVal = data.para, data.edgeInd, data
   .batchVal
        para = F.relu(self.convtn1(para, edgeInd))
        para = F.relu(self.convtn2(para, edgeInd))
        para = global_mean_pool(para, batchVal)
        return self.fcvar(para)


# Late Fusion Classifier
class LateFusionModel(nn.Module):
    def __init__(self , cnnOut=128, gcnOut=128, nClasses=124):
        super(LateFusionModel, self).__init__()
        self.cnn = CNNEncoder(cnn_out)
        self.gcn = GCNEncoder(gcn_out)
        self.classifier = nn.Linear(cnnOut + gcnOut, nClasses)

    def forward(self, image, graph_data):
        imgFeats = self.cnn(image)
        gcnFeats = self.gcn(graph_data)
        combo = torch.cat([imgFeats, gcnFeats], nDim=1)
        return self.classifier(combo)
```

A.3 Deep Reinforcement Learning Model

```
# 3. Set Dataset Path
DATASET_PATH = '/content/drive/MyDrive/Dataset'  # <<== Change
   this to your path

# 4. Load Silhouette Images
def load_images_universal(folder, target_size=(64, 64)):
    images = []
    labels = []
    label_to_idx = {}
    current_label_idx = 0

    def isImgFile(nameFile):
        return nameFile.lower().endswith(('.png', '.jpg', '.jpeg')
   )

    for root, dirs, files in os.walk(folder):
        imgFiles = [f for f in files if isImgFile(f)]
        if imgFiles:  # If there are image files in this folder
            folderClass = os.path.basename(root)
            if folderClass not in label_to_idx:
                label_to_idx[folderClass] = current_label_idx
                current_label_idx += 1
            for imgFile in imgFiles:
                imgLoc = os.path.join(root, imgFile)
                imgVar = cv2.imread(imgLoc, cv2.IMREAD_GRAYSCALE)
                if imgVar is None:
                    continue
                imgVar = cv2.resize(imgVar, target_size)
                images.append(imgVar)
                labels.append(label_to_idx[folderClass])

    return np.array(images), np.array(labels)


silhouetteImgs, gaitLabel = load_images_universal(DATASET_PATH)
print(silhouetteImgs.shape, gaitLabel.shape)
silhouetteImgs = silhouetteImgs / 255.0
silhouetteImgs = silhouetteImgs[..., np.newaxis]
print(f"Loaded {len(silhouetteImgs)} images!")

# 5. Feature Extractor CNN
class FeatureExtractor(nn.Module):
    def __init__(self):
        super(FeatureExtractor, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 32, 3, 1, 1), nn.ReLU(), nn.MaxPool2d(2),
            nn.Conv2d(32, 64, 3, 1, 1), nn.ReLU(), nn.MaxPool2d(2)
   ,
            nn.Flatten()
        )
        self.fc = nn.Linear(64 * 16 * 16, 128)

    def forward(self, para):
        para = self.conv(para)
        para = self.fc(para)
        return para

# 6. Dataset and Feature Extraction
class GaitDataset(nDataset):
    def __init__(self, imgs,  label):
        self.imgs = torch.tensor( imgs, dtype=torch.float32).
   permute(0, 3, 1, 2)
        self.label = torch.tensor( label, dtype=torch.long)
    def __len__(self):
        return len(self.imgs)

    def __getitem__(self, x):
        return self.imgs[x], self.label[x]

trainData = GaitDataset(silhouetteImgs, gaitLabel)
loader = DataLoader(trainData, batch_size=64, shuffle=False)

feature_extractor = FeatureExtractor()
feature_extractor.eval()

featureSeq = []
labelSeq = []

with torch.no_grad():
    for imgs, labels in loader:
        feats = feature_extractor(imgs)
        featureSeq.append(feats)
        labelSeq.append(labels)

features = torch.cat(featureSeq).numpy()
labels = torch.cat(labelSeq).numpy()
print(f"    Extracted {features.shape[0]} features of dimension {
   features.shape[1]}")

# 7. Define Gait Environment
class GaitEnv(gym.Env):
    def __init__( self, varFeatures, varLabels):
        super(GaitEnv, self).__init__()
        self.varFeatures = varFeatures
        self.varLabels = varLabels
        self.num_samples = len(varFeatures)
        self.currIndex = 0

        self.actionSpace = spaces.Discrete(len(np.unique(varLabels
   )))
        self.observationSpace = spaces.Box(lowVal=-np.inf, highVal
   =np.inf, shape=(varFeatures.shape[1],), dtype=np.float32)

    def resetFunc( self, seed=None, options=None):
        super().reset(seed=seed)
        self.currIndex = np.random.randint(0, self.num_samples)
        return self.features[self.currIndex], {}

    def step( self, action):
        correct_label = self.labels[self.currIndex]
        reward = 1 if action == correct_label else 0
        done = True
        info = {}
        next_obs = self.features[np.random.randint(0, self.
   num_samples)]
        return next_obs, reward, done, False, info


# 8. Train PPO Agent
env = GaitEnv(features, labels)
check_env(env)

model = PPO('MlpPolicy', env, verbose=1, batch_size=64,
   learning_rate=1e-4)
reward_log = []

for step in range(1, 11):  # Run for 10 epochs
    model.learn(total_timesteps=50000)
    correctValue = 0
    totalValue = 200
    for x in range(total):
        obser, x = env.reset()
        actionVar, x = model.predict(obser, deterministic=True)
        if actionVar == env.labels[env.currIndex]:
            correctValue += 1
    accuracyValue = correctValue / totalValue
    reward_log.append(accuracyValue)
    print(f"Epoch {step} - Test Accuracy: {accuracyValue:.4f}")


print(f"Final DRL PPO Test Accuracy: {correctValue/totalValue:.4f}
   ")
```