A skeleton-based study of gait with applications in lidar-based gait recognition and pathological gait identification

---

A Dissertation

Presented to

The faculty of the School of Engineering and Applied Science

University of Virginia

In partial fulfillment

of the requirements for the degree

Doctor of Philosophy (Electrical and Computer Engineering)

by

Nasrin Sadeghzadehyazdi

April 2021

Approval Sheet

This dissertation is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Electrical and Computer Engineering)

Author: Nasrin Sadeghzadehyazdi

This dissertation has been read and approved by the examining committee:

Scott T. Acton, Dissertation Adviser

Zongli Lin, Committee Chair

Laura Barnes, Committee Member

Tom Fletcher, Committee Member

Daniel S. Weller, Committee Member

Accepted for the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

April 2021

Acknowledgement

I would like to express my deepest gratitude to my advisor, Dr. Scott T. Acton for continuous support, encouragement, and guidance throughout my research work. My sincere thanks to my committee members, Dr. Zongli Lin, Dr. Laura Barnes, Dr. Dan Weller, and Dr. Tom Fletcher for their time and consideration.

I would like to thank all my wonderful friends in VIVA Laboratory for their help and support. I am truly grateful for the friendly atmosphere that I experienced during my PhD studies. My sincere thanks to Dr. Tamal Batabyal and Dr. Andrea Vaccari for their scientific advice and the insightful discussions about research.

A big thanks to my amazing friends, with whom I have shared so many cherished memories. I am fortunate to have your friendship and support.

To my parents and my siblings. Thank you for your endless love and support. Thank you for believing in me and encouraging me to pursue my dreams.

Lastly, to my husband, Hamid. Thank you for your patience, understanding, and support. I couldn't finish this journey without you.

Abstract

The study of human locomotion has been bolstered by automated gait analysis in the computer vision community. For years, gait analysis has been mostly limited to academic labs. The emergence of new modalities and the development of computational hardware that are essential for big data analysis has shifted gait analysis toward more practical methodologies. In recent years, gait analysis has emerged as a leading remote identification method for application in areas such as forensic investigation, surveillance, security, and medical fields.

Among the vision-based gait analysis methods, skeleton-based approaches are amenable for reliable feature compaction and fast processing. Model-based gait recognition methods that exploit features from a fitted model, like a skeleton, are recognized for their view and scale-invariant properties. This thesis investigates two problems associated with gait analysis: gait recognition and classification of gait abnormalities.

In the first part of this thesis, we focus on the application of flash lidar imagery to the gait recognition problem. Among available modalities, the emergence of depth cameras, such as Kinect and lidar that provide range (depth) and intensity simultaneously, has alleviated the computationally expensive model fitting that plays a critical role in many gait recognition studies. The current state-of-the-art model-based gait recognition methods take advantage of the high-quality data provided by Kinect and motion capture (Mocap) systems, which are mostly limited to controlled lab environments. Unlike Kinect and Mocap, the lidar camera is suitable for real-world applications; however, the data collected by lidar are noisy and have a lower associated resolution.

In this thesis, we utilize the data collected by a single flash lidar camera for the task of gait recognition. We seek to address the gait identification problem when a considerable number of feature vectors contain faulty and missing values. In particular, we will present methods to avoid the common practice of data elimination under the described conditions while still achieving high accuracy and precision in gait recognition. We describe filtering mechanisms to correct and interpolate the faulty and missing joint locations in the skeletons. In addition, methods are presented to incorporate the dynamic of the motion in the presence of noisy data. We discuss outlier removal as an alternative method for applications in which data elimination is not an issue and present a modification of Tukey's method for the vector-based attributes. Experimental evaluation demonstrates that joint

correction can effectively improve the classification scores in the proposed method and several relevant state-of-the-art approaches.

The second part of this thesis presents skeleton-based methods for the gait anomaly recognition problem. The main contributions in this part involve designing skeleton-based features and presenting end-to-end deep learning models that take minimally processed skeleton joints as the input. Unlike the common two-class or one-class approaches of skeleton-based methods, the proposed model considers a multi-class framework. Therefore, the approach can be easily adapted for a more convenient gait assessment outside clinical facilities. The proposed models are evaluated on three publicly available multi-class skeleton datasets with normal/pathological gait data, and achieve high classification scores in detecting minor gait abnormalities. The results indicate the potential of markerless modalities such as Kinect for designing less costly and more convenient health infrastructures for assisted living. Besides, an automatic and non-invasive gait assessment can further augment the clinical diagnosis for an extensive list of ailments that cause different types of gait disorders.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Gait analysis refers to the systematic study of the way that humans walk. Walking is an important aspect of daily life, yet we often ignore it due to its very nature of being a daily habit. Early studies in medicine and psychology have recognized the uniqueness of gait to individuals [6, 7]. Furthermore, it has been shown that gait can be affected by the mental and physical health of the subjects [8]. Gait has become an essential tool for identification and health evaluation. The development of modalities such as pressure sensors, video cameras, and accelerometers has allowed studying different aspects of the gait. Unlike other biometrics such as those computed from the face, fingerprint, and iris, gait can be acquired without the cooperation of the subjects, and contact between subjects and sensors is not required. These properties, along with the development of new modalities, have led to a widespread application of the gait analysis. In forensic studies, we use gait patterns to see whether the collected gait information in the crime scene matches with a certain individual [9]. In security monitoring and surveillance, video technology has been hired to recognize the potential threats by investigating the patterns of gait in suspicious individuals [10]. Researchers have been using gait for identity recognition [11–15], to recognize age and gender [16, 17], to detect abnormal behavior [18, 19], for human-robot

interfaces [20]. In medicine, gait analysis is employed for diagnosis of certain motion-affected diseases, or to evaluate the efficacy of therapeutic exercises [21–24].

Skeleton-based study of human gait describes a model-based framework for the gait analysis by way of fitting a skeleton to the human silhouette. This shift of modality from the structured (image/video) to unstructured (skeleton) datatype provides benefits in terms of data compaction, computation, storage, scalability, and recognition accuracy. The skeleton-related attributes mimic actual physical traits in the human body and can be utilized as a soft biometric identification (ID) for the individuals. Monitoring such physical traits over time also conveys valuable information about the health of an individual. The latter investigation has contributed to applications in assisted living, therapeutic, rehabilitation, sport, and medical field. This thesis exploits 3D skeleton data for applications in gait recognition and pathological gait identification. For gait recognition, we leverage pose estimation and focus on flash lidar modality. For normal/pathological gait identification, our main objective is to provide a pipeline for classification of minor gait anomalies that can be adopted for frequent gait monitoring outside clinical facilities.

In gait recognition, researchers study the gait features that can be utilized to identify individuals. Soft biometrics such as height, step length, and limb lengths have been derived as they mimic the actual physical traits of individuals [25–27]. The ultimate goal in gait recognition is to design gait features that act as a biometric ID.

The emergence of the depth cameras such as Kinect and lidar has provided the opportunity of investigating gait in the 3-dimensional real-world frame of reference. A depth-sensing camera generates intensity and depth (range) data simultaneously. The provided depth information is not affected by changes in illumination and lighting conditions, which are common issues with the intensity data in optical cameras. These properties make the depth-camera an ideal tool for gait analysis. Furthermore, with the direct estimation of skeleton joints by modalities such as Kinect, and deep learning-based

pose estimation tools, such as OpenPose [28] and DensePose [29], the computationally-expensive model fitting procedure, critical to model-based gait recognition methods, is not a costly task anymore.

In this thesis, we focus on the gait recognition problem through the lens of flash lidar imaging technology. A flash lidar is a proper choice for many real-world applications, and unlike Kinect and Mocap, it fits into a wide range of outdoor environment scenarios. However, the data collected by a flash lidar camera is low resolution and noisy that makes successful gait recognition a challenging task. With limited data availability, contrary to the common practice of noisy data elimination, we perform an extensive data correction to correct and recover noisy and missing skeleton joints extracted from the flash lidar data. Besides, to capture the dynamic of motion after data correction, we incorporate robust statistics to traditional feature moments.

The second part of this thesis investigates the application of gait analysis for the detection of gait abnormalities. Numerous studies have shown the significance of gait in the health assessment of individuals [30, 31]. Researchers have found that the patterns of gait can be adopted for diagnosis and severity detection in various neurological and physiological ailments [32, 33]. Furthermore, analysis of gait patterns is an essential tool in rehabilitation after injuries or a surgery [34, 35]. Clinical assessments of gait are conducted in specialized lab facilities, which are generally costly. In addition, the application of essential tools often requires careful sensor placement that is time-consuming and inconvenient. More recently, wearable sensors have become more common in health-related studies of gait [36, 37]. In practice, these sensors are light-weight and less costly. Nonetheless, the quality of the generated data is affected by the correct placement of the sensors [38]. Multiple studies have investigated the liability of Microsoft Kinect as a modality for gait evaluation [39, 40]. Unlike wearable sensors, Kinect does not require

any accurate sensor placement. The main advantages of Kinect are its low cost and accurate markerless skeleton joint detection and tracking. The latter property makes the application of Kinect convenient for both patients and the healthcare provider.

The majority of state-of-the-art skeleton-based gait anomaly recognition studies are categorized as either a one-class or a two-class problem, recognizing anomalous from normal class only. Numerous of these works have reported high recognition scores. However, in the real world, abnormal gait can be divided into multiple categories, each describing one symptom or a minor abnormality. In this study, we take a multi-class approach toward gait anomaly classification based on the skeleton modality. Under a multi-class framework, gait anomaly recognition can be adopted for real-world applications, which can render frequent gait evaluation out of designated lab facilities. The presented work offers an end-to-end method toward gait anomaly classification, where minimally preprocessed skeleton joints are fed to a deep learning model. By taking an end-to-end approach, we let the deep model detect relevant features by itself while attempting to minimize classification error over all existing classes. Also, unlike other skeleton-based gait anomaly recognition, we evaluate the presented models on three public datasets with different gait abnormalities. High classification scores on these datasets with distinct classes of pathological abnormalities have two main indications. It shows the efficiency of the presented pipeline in classifying minor gait abnormalities. Besides, it shows the usefulness of markerless modalities such as the Kinect camera for minor gait anomaly classification.

1.1 Objectives and contributions

The primary objective of this thesis is to design efficacious pipelines based on 3D skeleton data for applications in gait identification and gait anomaly classification, by leveraging

depth-based cameras such as flash lidar and Kinect. To deliver on such objectives, we attempt to address the following questions:

- If the collected data are noisy to a level such that a considerable number of feature vectors contain faulty and missing values, can we still achieve high accuracy and precision in gait recognition?
- When a high percentage of the input features are either noisy or missing, can we avoid data elimination and do any better through model correction?
- Can we perform frequent gait evaluation in the convenience of living environment with a low-cost and markerless equipment such as Kinect and without the need for an equipped specialized lab?

The first two questions are investigated in the first part of this thesis, as we describe a simple yet effective pipeline for gait recognition designed for flash lidar modality. We will address the last question in the second part of this thesis, where we present a pipeline for multi-class gait anomaly recognition. The following list outlines the contributions as we address each of these questions:

Contribution 1:

The main goal in the first part of this thesis is to present a pipeline for gait identification that overcomes the difficulties that arise as a result of imaging with flash lidar. In order to fulfill this goal, we adopt a model-based gait recognition approach due to the view and scale-invariant properties of this type of method. In addition, a model-free method generally requires background removal that is quite challenging with the low resolution and noisy data provided by the flash lidar. To remove the computationally expensive process of three-dimensional model fitting, we use the depth data provided by lidar and hire a pretrained pose detector model.

The noisy nature of lidar data presents a real challenge to the skeleton detection procedure, degrading the performance of the state-of-the-art pose detectors. Therefore, erroneous and missing joint location measurements is a real issue, resulting in gait recognition with low precision and accuracy. To resolve this problem and improve the gait identification scores, we will discuss offline approaches to correct the faulty skeleton joints. Correction is performed on each joint location coordinate, modeled as a time sequence. With limited data availability, data correction is valuable as it preserves temporal information, which is critical for timely applications such as gait recognition. To capture motion dynamics after an extensive skeleton correction, we incorporate robust statistics with traditional feature moments. Through a set of experiments, we show that conventional feature moments can be a better representative of motion dynamics if they are incorporated with robust statistics such as median, lower and upper quartiles.

We will also discuss performing outlier removal on the feature vectors to acquire higher quality data and present methods for length-based and vector-based attributes. The presented outlier removal methods can be adopted for applications where data elimination is not an issue. This contribution is an effort to follow the traditional practice of removing noisy data and performing classification on the remaining higher quality examples.

Contribution 2:

In the second part of this thesis, we focus on the problem of anomaly recognition with the skeleton data collected by Kinect. The majority of state-of-the-art gait anomaly detection methods have been focused on evaluating certain parameters of the gait that are clinically relevant. Such analysis is valuable, as they link health evaluation of gait with suitable parameters that are clinically interpretable. However, they also limit the acquired information to a few selected factors and do not take into account the interaction between different body segments in forming gait patterns. Skeleton-based models try to

avoid the limitation in traditional approaches by taking into account the interactions between individual body parts.

We purpose an end-to-end deep learning model that uses the skeleton data recorded by Kinect to capture spatio-temporal patterns for gait anomaly recognition. By considering the whole skeleton, the proposed model considers the relationship between different body joints in locomotion. Unlike the common two-class or one-class approaches of skeleton-based methods, the proposed model considers a multi-class framework. Therefore, it can be easily adapted for a more frequent gait assessment outside clinical facilities. Besides, for the first time, we evaluate the purposed pipeline on the three largest publicly available datasets. The high classification scores that are acquired on all three datasets demonstrate the efficacy of the proposed pipeline for minor gait anomaly recognition. Also, the whole framework of this design indicates the potential of markerless modalities such as Kinect for designing less costly and more convenient health infrastructures for assisted living. In addition, an automatic and non-invasive gait assessment can further augment the clinical diagnosis for an extensive list of ailments that cause different types of gait disorders.

1.2 Thesis outline

The dissertation is organized as follows: In Chapter 2, we present some background on gait recognition, describe the challenges of gait recognition with flash lidar, and present our methodology for gait recognition in the framework of flash lidar modality. Chapter 3 presents methods for improving the gait recognition methodology with the flash lidar data. Chapter 4 is dedicated to the problem of gait anomaly recognition. In this chapter, we present methods based on deep learning models for classification of minor gait anomalies using the skeleton data. Finally, we conclude in Chapter 5 and outline avenues for future work.

Chapter 2

Gait recognition with flash lidar

In this chapter, first, we present a brief background on the gait recognition problem and outline the advantages of using a depth-based camera, such as flash lidar, for gait recognition. Next, we explain the imaging mechanism of the flash lidar camera and describe the properties of the collected data. We present a model-based gait recognition method for the data collected by a single flash lidar camera. Finally, we will present the experimental results and outline some of the challenges of gait recognition with the flash lidar data.

2.0.1 Background on gait recognition

Traditionally, there are two dominant trends in video-based gait recognition, model-based and model-free methods. Model-free methods utilize the features that are computed from human silhouette [41,42]. In terms of implementation and computation, model-free methods are less costly compared to their model-based counterparts. However, the performance of model-free approaches depends on the quality of silhouettes. The silhouette quality is affected by several factors such as lighting conditions and outfits of the subjects [43, 44]. Model-based methods exploit the features that can be computed from a fitted model, like a skeleton [45, 46]. Therefore, this class of methods is scale and view-invariant. While

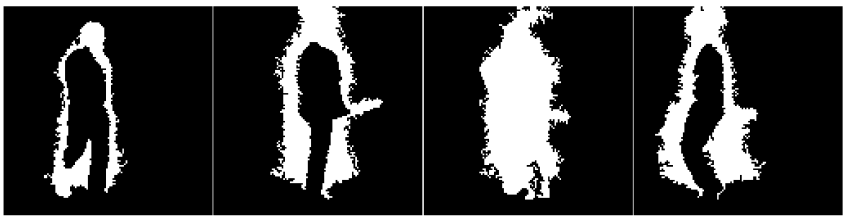


FIGURE 2.1: Examples of noisy segmented silhouettes from flash lidar data

model-free methods deliver a representation of human shape, model-based approaches describe human locomotion. Skeleton-based features mimic actual physical traits, and the shift of modality from image or video to skeleton offers benefits in terms of compaction, storage, computation, and scalability. With the emergence of depth cameras like Kinect and lidar, the computationally expensive process of model fitting is not an issue anymore.

The only existing lidar-based gait recognition studies are model-free, rely on the data provided by rotating multi-beam lidar, and exploit point clouds to extract the silhouette [42, 47]. With the data that is collected by a single flash lidar camera, several factors diminish the quality of segmented silhouettes. Figure 2.1 shows examples of faulty detected silhouettes. Apart from faulty silhouettes, there are frames with no detected silhouette, which mostly happen in successive frames.

There are a few studies in the literature that address the problem of gait identification with low quality or missing silhouettes. In [48] and [49], the authors study multiple scenarios with incomplete silhouettes. But, these researchers do not address the cases when an entire silhouette is missing. In general, these studies depend on the proper segmentation of a reference silhouette. Silhouette reconstruction methods such as inpainting are only effective when smaller parts of the silhouette are missing [49]. Methods based on gait features such as gait energy image (GEI) [11] and its variations, which are less sensitive to segmentation error, are also based on the non-missing silhouette criterion. In [50] and [51], the problem of the missing silhouette is treated. However, these studies only focus on sequences with a 90-degree camera view in the former and frontal view in

the latter study. A 3D model-based approach is view and scale-invariant and can avoid the problem of missing and faulty segmented silhouettes.

Multiple studies have utilized skeleton joint information provided by Kinect in applications such as activity recognition, person identification, and gait analysis [52–54]. Within the framework of gait recognition, authors have investigated angle-based attributes [25], static anthropometric [55], and gait features [56]. In [26], authors took covariance-based features of skeleton joints' trajectory. Sinha *et al.* combined a set of area-based features and the distance between body segment centroids with the angle between lower body limbs and anthropometric attributes [57]. In [58], relative distance and angles were used along with the Dynamic Time Warping (DTW). Ali *et al.* introduced area-based features of the lower body during motion [59]. Weighted anthropometric, dynamic, and trajectory features over segmented gait cycles were presented in [20]. In the majority of these studies, the mean, maximum, and standard deviation of the proposed features over each gait cycle integrate the motion dynamic for a high-accuracy recognition. A gait cycle is a fundamental concept in describing the human locomotion and is defined with respect to one of the legs. One gait cycle is the time between an initial contact between one foot and the ground and the next contact of the same foot with the ground.

In this dissertation, we take a model-based approach for the gait recognition problem using the data recorded by a single flash lidar camera. To tackle the computationally expensive model fitting problem, we hire a pretrained deep network for pose detection.

2.1 The proposed model-based method

The input to the proposed gait recognition system are sequences recorded by a flash lidar camera. A lidar sequence like V with f frames, consists of intensity $I = [I_1, I_2, ..., I_f]$ and range (depth) frames $R = [R_1, R_2, ..., R_f]$, where the images are preprocessed to reduce

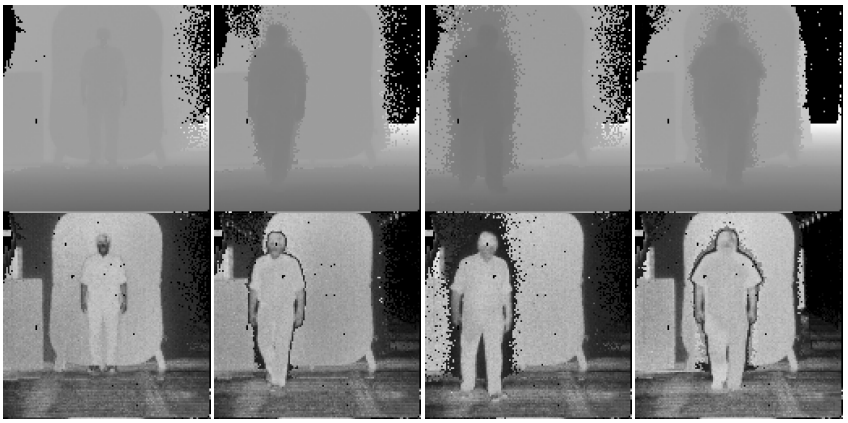


FIGURE 2.2: Sample frames of lidar data. The top and bottom rows show range (depth) and intensity data, respectively.

the noise in the data. Sample frames of intensity and range data are shown in Figure 2.2, top and bottom rows, respectively. Using the intensity information of lidar, we leverage OpenPose, a state-of-the-art real-time pose detector [28] to fit a two-dimensional skeleton model, and extract the location of the body joints. Figure 2.3 illustrates sample frames with correctly detected skeletons. The two-dimensional skeletons generated by OpenPose has 18 joints. However, 5 out of 18 joints represent facial limbs that do not convey any information about gait. By removing these 5 joints, we adopt a skeleton model that contains the remaining 13 joints. Figure 2.4 gives an illustration of the skeleton model that we use in this work. The table on the left side of this figure lists all the joints in this model.

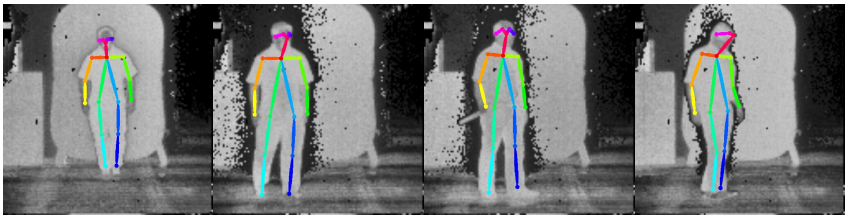


FIGURE 2.3: Sample frames with correctly detected skeletons.

With I_i being the input to the skeleton detector, the output is the joint location coordinates J_i in the following vectorized form

$$J_i = [x_k, y_k]_{k=1}^{M_j} \in \Re^{2N} \tag{2.1}$$

Index	Joint
1	Mid Shoulder
2	Right Shoulder
3	Right Elbow
4	Right Wrist
5	Left Shoulder
6	Left Elbow
7	Left Wrist
8	Right Hip
9	Right Knee
10	Right Ankle
11	Left Hip
12	Left Knee
13	Left Ankle

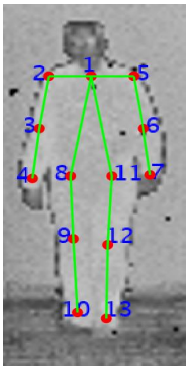


FIGURE 2.4: The skeleton model we use in this work. Left: index of each joint in the skeleton model. Right: skeleton model in a sample frame.

where M_j is the number of joints and (x_k, y_k) represents the coordinates of the kth joint in the image frame of reference. The 2-dimensional coordinates of the joints in the x and y directions are projected into real-world coordinates using the range data and the properties of the lidar camera. Equation 2.2 describes the projection from image reference frame into the real-world coordinates system

$$L_j^i = \frac{2}{N_{pixels}} \times \tan(\frac{\theta_{aov}}{2}) \times Lp_j^i \times D_{camera}^i \tag{2.2}$$

where L_j^i represents the real-world location of joint i in the direction $j \in \{x, y\}$. L^i in the z direction equals to the depth (range) value at the location of joint i. N_{pixels} is the number of pixels in the j direction, θ_{aov} represents the angle of view, and D_{camera}^i is the range value of joint i. Lp_j^i shows the location of joint i in the direction j in the image coordinate system.

2.1.1 Feature vectors

In this work, the purposed methods are tested on two sets of feature vectors: length-based (LB) and vector-based (VB). The length-based feature vector comprises a set of limb lengths and Euclidean distance between selected joints in the skeleton. Table 2.1 lists the components of the length-based feature vector.

TABLE 2.1: List of length-based feature vectors (L and R refer to the left and right joints, respectively)

Feature	**Feature**
R and L Shoulder	Elbow to elbow
R and L upper arm	Wrist to wrist
R and L lower arm	Hip to hip
Spine	Knee to knee
R and L upper leg	Ankle to ankle
R and L lower leg	R shoulder to L ankle
Shoulder to shoulder	L shoulder to R ankle

TABLE 2.2: List of three-dimensional vectors in the vector-based feature vector (L and R refer to the left and right joints, respectively)

Feature	**Feature**
Neck to R Shoulder	R Hip to R Knee
Neck to L Shoulder	L Hip to L Knee
Neck to R Hip	R Elbow to R Wrist
Neck to L Hip	L Elbow to L Wrist
R Shoulder to R Elbow	R Knee to R Ankle
L Shoulder to L Elbow	L Knee to L Ankle

The second set of feature vectors is vector-based. This means that each feature is a 3-dimensional vector, computed between two skeleton joints. Unlike trajectory features in [26] that are computed with respect to a reference joint, the vectors in the proposed feature vector mimic the limb vectors in the skeleton model. Vector-based features provide information about the angle and distance between selected joints of the skeleton. In Table 2.2 we list the joints that form each of the 3-dimensional vectors. Figure 2.5 presents an illustration of the proposed length-based and vector-based features.

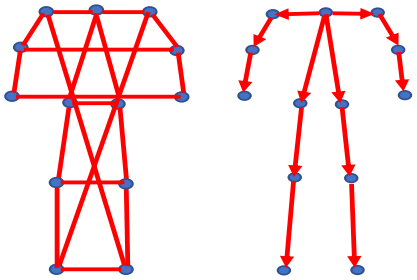


FIGURE 2.5: Illustration of two types of feature vectors: distance-based feature vector (left), vector-based feature vector (right). All The features are depicted in red color.

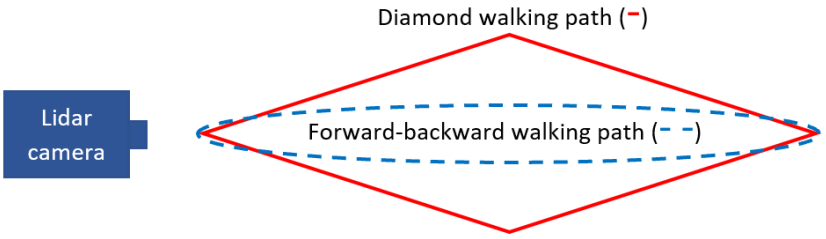


FIGURE 2.6: Illustration of two types of walking path: walking forward and backward (in blue), and diamond walking (in red).

2.1.2 TigerCub 3D Flash lidar

The TigerCub is a light-weight 3D flash lidar camera that uses eye-safe pulsed laser to illuminate the whole scene and generates real-time range and intensity data [60]. A laser beam can be focused to conform to the objects of interest. Therefore, a lidar camera can provide a detailed depth imaging of the recorded scene. These properties of flash lidar have lead to extensive applications in areas such as geology, seismology, atmospheric physics, forestry, archaeology, autonomous driving, and space missions. The capability of the lidar camera to perform robustly in the dark, in the fog, and the dust, makes it stand out among other depth-based cameras. The working range of a flash lidar camera is above 1000 meters, and in the generated detailed 3D mapping, the spatially close objects can be recognized from one another. Considering such properties, flash lidar can be a suitable candidate for real-time data acquisition and autonomous operations.

2.1.3 Dataset

The dataset in this work has been collected by a single TigerCub 3D Flash lidar camera. The data is captured at the rate of 15 fps with 128×128 frame resolution. The dataset consists of a total of 34 sequences of the walking action performed by 10 subjects. Each subject performs the walking action in three different ways: walking toward and away from the camera, walking on a diamond shape, and walking on a diamond shape while holding a yardstick with one hand. Figure 2.6 illustrates the paths of walking for the two

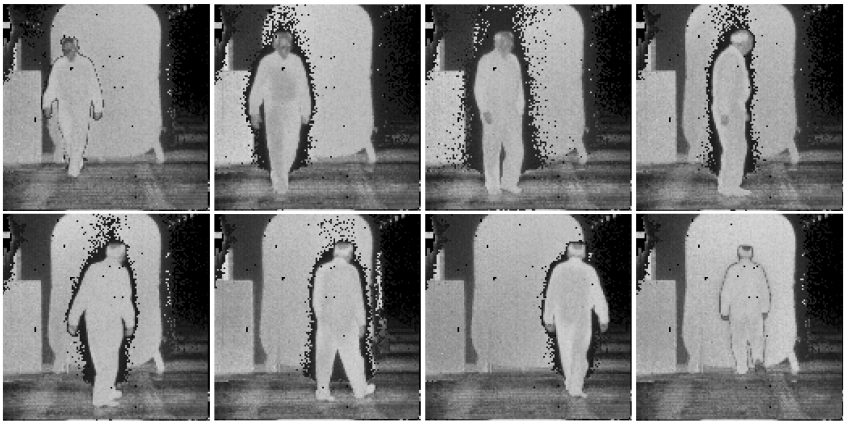


FIGURE 2.7: Sample frames of diamond walking that captures a range of different poses.

cases of walking forward and backward (walking toward and away from the camera) and the diamond walking. For those frames in which subjects walk toward and away from the camera, most of the views are from the front and back of the person, with some frames of side views when the subjects turn away. The sequences with walking on a diamond shape include frames with a wider range of views. This will offer a wider range of poses as is shown in Figure 2.7.

Table 2.3 lists the number of frames per subject for each category of the walking action. The number of frames per video is different, ranging from 130 to 498 frames. Each frame has two sets of data, intensity and range, both with the same number of pixels. The intensity data are presented in gray-scale, and the range data show the distance of each point in the field of view from the camera sensor.

TABLE 2.3: Number of frames per type of walking action for each subject. FB Walk: front back walk, D Walk: diamond walk, DS Walk: diamond walk holding stick

	FB Walk	D Walk	DS Walk	Total
subject 1	130	215	463	808
subject 2	248	462	451	1161
subject 3	199	398	391	988
subject 4	224	377	405	1006
subject 5	257	459	486	1202
subject 6	226	483	881	1590
subject 7	204	429	394	1027
subject 8	249	474	445	1168
subject 9	203	897	375	1475
subject 10	216	441	385	1042

2.1.4 Experimental results

Table 2.4 summarizes the average accuracy and F-score for the proposed features and several relevant methods. In [56], authors use a set of static features plus step length and speed. In [25], the moments of six lower body angles are computed over each gait cycle. Sinha *et al.* integrate the features in [56] and [25] with their area-based and distance between body segment features and compute the moments of each feature over every gait cycle. Instead of the common anthropometric features, in [27], Yang *et al.* utilize selected relative distance along different motion directions. As can be seen from the results, the proposed vector-based feature outperforms the state-of-the-art related methods. We also observe that none of the methods could acquire high classification scores as a result of low quality skeletonization and the resulting erroneous features.

TABLE 2.4: Correct identification scores for the proposed features and the other methods. LB and VB stand for length-based and vector-based feature vector, respectively. Features are computed without joint correction.

Method	**Average Accuracy(%)**	**Average F-score(%)**
[56]	27.90	25.36
[25]	25.34	23.24
[57]	61.81	54.61
[27]	63.82	58.64
Ours, LB	54.96	51.58
Ours, VB	<u>67.16</u>	<u>63.47</u>

Several factors diminish the quality of the joint localization, and therefore the features that are computed from the skeleton's joints. By looking at the sample intensity frames in Figure 2.2, bottom row, we observe the lack of color, and similarity between the clothing of the subjects, skin, and the background. Depth images are plagued with edge noise and missing pixels. Furthermore, as the distance between subjects and the camera lessens, range data is affected by noise. The acquired skeletons are riddled with missing and inaccurately-located joints. Therefore, the resulting features contain many missing and noisy measurements.

2.2 Remarks

In this chapter, we described the gait recognition problem and addressed some of the challenges and opportunities of gait recognition with the flash lidar data. We discussed a gait recognition methodology for the flash lidar modality by presenting skeleton-based features. Our results and analysis shows how performing a successful gait identification using the flash lidar data is a challenging task. In the next chapter, we will come back to this problem and present methods for improving gait recognition for the data collected by a flash lidar camera.

Chapter 3

Improving flash lidar-based gait identification

Most of the existing successful model-based methods rely on the high-quality data collected by Kinect or Mocap. While such modalities have removed the burden of model fitting, they are not always the best choice for real-world applications. Mocap is mostly limited to laboratory environments. The working range of the Kinect is very limited (< 5 meters) and its performance degrades in outdoor environments because the infrared light of the sensor cannot be easily separated from the high-intensity ambient infrared [61, 62]. On the other hand, flash lidar has an extensive working range (> 1000 meters). Furthermore, due to the high irradiance power of pulsed laser with respect to the background, the performance of a flash lidar is not degraded in the outdoor environments. However, as we discussed in the previous chapter, several factors diminish the quality of the data collected by a flash lidar camera. Therefore, compared with Kinect and Mocap, the data collected by flash lidar is noisy and low resolution. These factors degrade the performance of the pose detector, resulting in many missing and faulty joint localization.

Under the described condition, a common practice consists of noisy data removal and performing further processing on the remaining clean data. In this chapter, following the

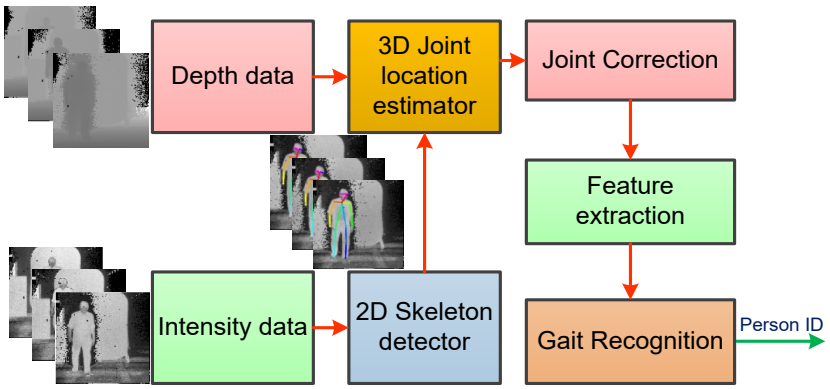


FIGURE 3.1: Pipeline for gait recognition using the joint correction methodology

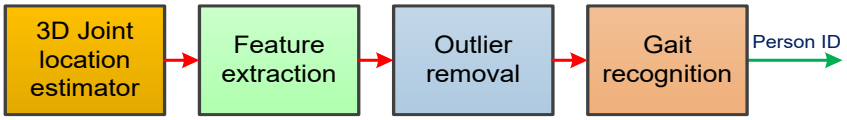


FIGURE 3.2: Pipeline for gait recognition using the outlier removal methodology. Inputs to *"3D Joint location estimator"* remain the same as in Figure 3.1

traditional trend of noisy data removal, we employ the Tukey method for outlier removal and present a modification for the vector-based features. This approach results in a higher quality data; however, it might not be the best choice for real-world surveillance problems with limited data availability. In fact, under such a scenario, data elimination can be problematic. We will address this problem and present filtering methods to correct the noisy joints in the time sequences of joint coordinate measurements. Furthermore, robust statistics are integrated with conventional feature moments to encode the dynamics of the motion after skeleton joint correction.

Figures 3.1 and 3.2 describe the workflow for gait recognition based on joint coordinate correction and outlier removal, respectively. In the following sections, we will describe the steps in each of these two pipelines and present an extensive set of experiments to investigate the efficacy of the proposed methods.

3.1 Outlier removal

Outlier samples do not follow the underlying model of a process. In general, outliers should be detected to either understand an interesting event or process (i.e. surveillance and abnormal behavior) or be removed if they are the result of noise or caused by erroneous measurements. A model that is estimated based on a dataset corrupted by such outliers, cannot provide a fair description of the system and will result in many false predictions. A common practice in gait recognition involves removing outliers by setting some thresholds and performing the main analysis on the remaining higher quality data [20, 27, 63, 64]. Therefore, as an alternative method, we use the Tukey method to detect outliers in the feature vectors that are computed from noisy and missing joint location coordinates. The pipeline for gait recognition based on outlier removal methodology is given in Figure 3.2. By choosing the Tukey method, we avoid making any assumption about the underlying distribution of each feature. The second row in Figure 3.3 shows examples of faulty detected skeletons. Besides, there are frames with missing skeletons.

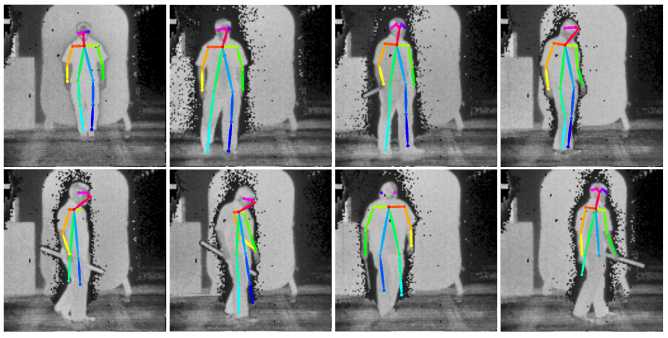


FIGURE 3.3: Top row: sample frames with correctly detected skeletons, bottom row: frames with faulty skeletons

3.1.1 Outlier removal for length-based features

We define $Jd = [Jd_1, Jd_2, ..., Jd_P]$ as a feature vector, where P is the number of features in Jd and Jd_i is the Euclidean distance between two skeleton joints. For length-based features, removing outliers is performed in three main steps. First, we remove all the

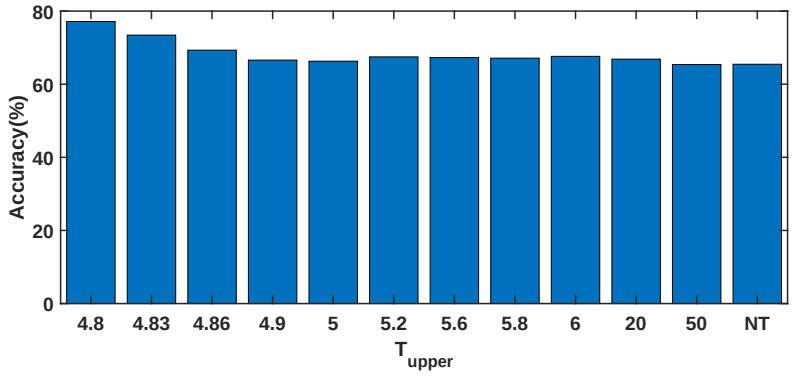


FIGURE 3.4: Average classification accuracy with length-based outlier removal for different values of T_{upper} (threshold value that is used for prefiltering in length-based outlier removal). $T_{upper} = NT$ means no threshold was applied. While $T_{upper} = 4.8$ results in the highest accuracy, it also results in the lowest percentage of training and test data being preserved after applying Tukey's test. Higher values of T_{upper} or no threshold preserve more than twice the number of training and test samples compared with $T_{upper} = 4.8$.

frames with missing skeletons. In the second step, we filter the remaining samples by setting an upper threshold of T_{upper}. To determine T_{upper}, we compute the median and interquartile of each feature in Jd

$$M_{Jd} = max(median(Jd_i) + IQR(Jd_i))|_{i=1}^{P} \tag{3.1}$$

If Jd^s is the feature that maximizes the summation in the above equation, then M_{Jd} is the value of the above summation for feature Jd^s. Once we determine M_{Jd}, we perform a grid search around the value of M_{Jd} to find T_{upper}. T_{upper} is the value that results in the highest accuracy after applying Tukey's test in the next step.

A feature vector with a feature that is beyond T_{upper} will be removed. In the last step, Tukey's test is employed on each feature. Jd is not an outlier if

$$Tukey(\{Jd_i\}_{i=1}^{P}) = \mathbf{0}_P \qquad where \qquad Jd_i \in \Re^+ \tag{3.2}$$

where $\mathbf{0}_P$ is a zero vector of length P. $Tukey(Jd_i) = 0$ means that feature Jd_i passed the Tukey's test, or Jd_i is not an outlier. Based on Equation 3.2, Jd is not an outlier, if all of its feature components are non-outliers. In other words, Jd is an outlier if there

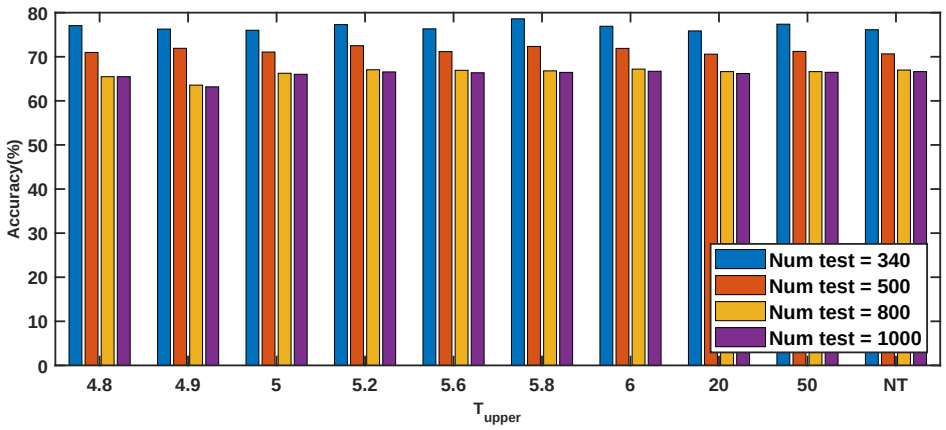


FIGURE 3.5: Test accuracy for length-based outlier removal for different threshold values (T_{upper}), given different numbers of test samples. $T_{upper} = NT$ means no threshold was applied. While smaller values of T_{upper} result in higher classification accuracy for a smaller number of test samples, this difference disappears for larger numbers of test samples ($number\ of\ test\ samples >= 800$)

exists a Jd_i, such that $Tukey(Jd_i) = 1$. We will show later that while outlier removal will improve gait recognition scores, it comes at the cost of eliminating a considerable portion of the data.

Figure 3.4 shows the test accuracy with different values of T_{upper}. Smaller values of T_{upper} result in the removal of one or more classes and therefore are not included in this figure. In Figure 3.4, the distance between juxtaposed T_{upper} values (values along x-axis) become larger as we go toward larger values. This is because those values in-between the shown T_{upper}s did not make a difference in terms of accuracy compared with values close to them that are illustrated in this figure. $T_{upper} = NT$ means **n**o **t**hreshold was applied for prefiltering, and only the samples with no skeleton have been removed before applying Tukey's method. It is important to keep in mind that the value of T_{upper} that gives the highest accuracy also preserves the smallest percentage of training and test data after applying Tukey's test. In contrast, using higher values of T_{upper} or removing T_{upper} altogether ($T_{upper} = NT$) corresponds with preserving a higher percentage of the training and test data.

Figure 3.5 shows a comparison of the performance of length-based outlier removal for different values of T_{upper}, given various numbers of test samples. The comparison

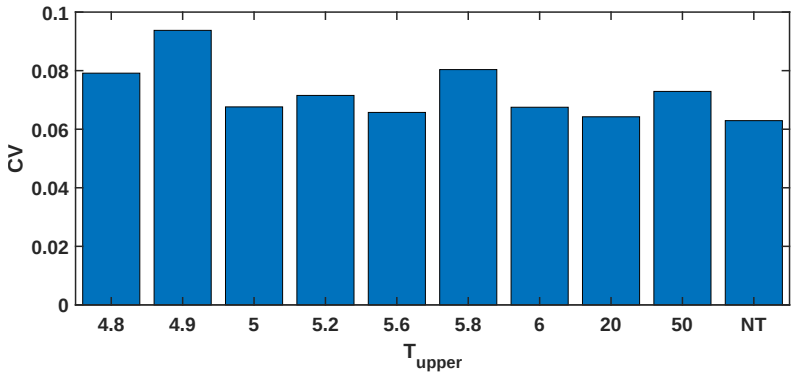


FIGURE 3.6: CV (coefficient of variation) of average classification accuracy over different numbers of test samples for various values of T_{upper} (threshold value that is used for prefiltering in length-based outlier removal). $T_{upper} = NT$ means no threshold was applied. The lowest CV, which is achieved with $T_{upper} = NT$, results in the least dispersion of accuracy over different numbers of test samples.

shows that for smaller numbers of test samples, values of T_{upper} close to M_{Jd} give higher classification accuracy. But, as we increase the number of test samples, this difference fades. To study the effect of T_{upper}, for each value of T_{upper}, we also calculate the coefficient of variation (CV) of the test accuracy over different numbers of test samples. For each T_{upper}, we compute accuracy for different numbers of test samples and then compute the CV of the resulting test accuracy values for the given T_{upper}. $CV = \sigma/\mu$, where σ and μ are defined as standard deviation and mean of the resulting test accuracy values, respectively. As we observe in Figure 3.6, the case with no threshold ($T_{upper} = NT$) gives the lowest CV. These results can indicate that applying no threshold creates a more reliable classifier with the least percentage of accuracy dispersion, given various numbers of test samples.

3.1.2 Outlier removal for vector-based features

Let $Jv^{3D} = [Jv_1^{3D}, Jv_2^{3D}, ..., Jv_Q^{3D}]$ be a $3 \times Q$ matrix of the joint coordinates, where Q is the number of 3-dimensional vectors in Jv^{3D}. The ith column in Jv^{3D}, which is the 3-dimensional vector between two skeleton joints, is defined as follows

$$Jv_i^{3D} = [x_i, y_i, z_i] \in \Re^{3N} \tag{3.3}$$

Each of the 3-dimensional vectors represents one entity and cannot be treated individually. To reduce each 3-dimensional vector to one entity, we first use the concept of marginal median [65]. Each component of the marginal median represents the median of all the vector components in that direction. Next, we employ cosine distance to calculate vector similarity between each set of 3-dimensional vectors with their corresponding median vector. We define Jv^{median} as the marginal median over all the given Jv^{3D} feature vectors

$$S^{3D} = cos(Jv_i^{median}, Jv_i^{3D})|_{i=1}^{Q} \tag{3.4}$$

where $S_i^{3D} = cos(Jv_i^{median}, Jv_i^{3D})$ is the cosine similarity between ith element of feature vector Jv^{3D} and Jv^{median}. This way, we create the cosine similarity measure between each Jv^{3D} and the median vector Jv^{median} and reduce each 3-dimensional vector in Jv^{3D} to one entity. Lastly, the Tukey method is applied to the cosine similarity measures. A feature vector is an outlier if at least one of its features is an outlier. The algorithm below describes outlier detection on the feature vectors built from 3-dimensional vectors.

Outlier detection for vector-based features

1. Over all the given feature vectors, calculate the marginal median vector. Call this vector Jv^{median}

2. For each 3D vector Jv_i^{3D} in each feature vector Jv^{3D}, calculate $cos(Jv_i^{median}, Jv_i^{3D})$; save the results in one row of S.

3. Employ Tukey's test on each row of S.

4. A given feature vector Jv^{3D} will pass Tukey's test if its corresponding row in S passes Tukey's test.

3.1.3 Experimental results with outlier removal

Table 3.1 summarizes the gait identification scores before and after applying outlier removal. We observe that applying outlier removal can improve the identification scores for both types of features. However, outlier removal also results in the elimination of the data that can be problematic when data is limited. Due to the noisy and low quality nature of flash lidar data, a high percentage of the detected skeletons have missing and noisy joints. For such cases, outlier removal can eliminate a good portion of the collected data. Besides, data elimination will result in the loss of temporal information that is valuable for applications such as gait identification and activity recognition. In the following section, we describe skeleton joint location correction to resolve these problems.

TABLE 3.1: Correct identification scores for the proposed features before and after applying outlier removal. LB and VB stand for the length-based and vector-based feature vectors, respectively.

Method	**Average Accuracy(%)**	**Average F-score(%)**
LB (before)	54.96	51.58
VB (before)	67.16	63.47
LB (after)	76.60	68.89
VB (after)	<u>80.70</u>	<u>75.22</u>

3.2 Skeleton joint correction

To improve gait recognition, we present two filtering mechanisms. First we describe GlidarCo (**g**ait recognition by **lidar** through joint **co**rrection). GlidarCo is a three-step filtering mechanism that corrects erroneous joint location measurements and recovers missing joints. Next, we describe GlidarPoly (**g**ait recognition by **lidar** through **poly**nomial correction), that is a two-step filtering approach. Besides, we will show that with an extensive skeleton correction, we can improve gait identification even further by incorporating robust statistics with the common feature moments.

3.2.1 Skeleton joint correction by GlidarCo

Figure 3.1 illustrates the joint correction methodology. By investigating the time sequences of the joint location coordinates, we realize that missing joint location measurements form the majority of erroneous joint localization. In order to perform joint correction, we model each joint location of a skeleton in a lidar video as a time sequence, where each joint location is composed of three time sequences in the x, y, and z directions. Given a skeleton model of 13 joints, we define L as a matrix of the size of $39 \times F_n$, where each row represents one time sequence that is extended over F_n frames. To correct the joint localization, filtering is carried out on each row of the L matrix, where L_m represents the mth row of L

$$L_m = \{L_m(t)\}_{t=1}^{F_n} \qquad L_m(t) \in \Re \tag{3.5}$$

For each row like L_m, we find the sorted location of all the nonzero elements. For each lidar video, given matrix L, we perform filtering on L_m (each row of L), in three main steps. First, Tukey's test is utilized to detect all the values that are below $Qu_{low} - 1.5 \times IQR$ or above $Qu_{up} + 1.5 \times IQR$, where $IQR = Qu_{up} - Qu_{low}$ stands for the interquartile range, Qu_{low} and Qu_{up} are lower and upper quartile or 25 and 75 percentiles, respectively. Defining o_{L_m} as the set of all the detected outlier indices in L_m (each index corresponds with one frame)

$$\begin{cases} o_{L_m} = [o_1, o_2, ..., o_R] \\ o_1 < o_2 < ... < o_R \\ o_{L_m} \in [1, ..., F_n] \end{cases} \tag{3.6}$$

where R is the number of detected outliers in L_m. Each detected outlier will be corrected by the value of its one nearest neighbor in time that is not an outlier. In those cases with two nearest neighbors, one is selected randomly. In the second step, piece-wise cubic Hermite polynomials [66] are utilized to interpolate the missing values in L_m over

(t_k, t_{k+1}) interval

$$P(t) = \begin{bmatrix} 2t^3 - 3t^2 + 1 \\ -2t^3 + 3t^2 \\ t^3 - 2t^3 + t \\ t^3 - t^2 \end{bmatrix} \begin{bmatrix} t_k \\ t_{k+1} \\ \nabla t_k \\ \nabla t_{k+1} \end{bmatrix} \tag{3.7}$$

where ∇t_k and ∇t_{k+1} are the derivative at t_k and t_{k+1}, respectively.

In the final step, RLOWESS (locally weighted scattered plot smoothing) [67] is performed to smooth the resulting curve of L_m, and mitigate the effect of the remaining outlier values that resemble lower-amplitude spikes. RLOWESS locally fits first order polynomial using weighted linear regression. In the first step, the weighted least square problem in each neighborhood is solved using the tricube weight function that is defined according to the following equation

$$w_i = (1 - \left| \frac{t - t_i}{d(t)} \right|^3)^3 \tag{3.8}$$

where w_i is the regression weight of point t_i located in the neighborhood of t, and $d(t)$ is the distance along the abscissa between t and the point which is furthest from t in its designated neighborhood (t and t_i are the predictor values). Using these weights, each point like t is estimated and the residual of that point calculated $r = t - \hat{t}$, where $\hat{t}$ is the estimation of t through the weighted least square. Next, the robust weight of point t_i is calculated by the bisquare weight function

$$w_i = (1 - \left| \frac{r_i}{6 \times median(|r|)} \right|^2)^2 \tag{3.9}$$

where $median(|r|)$ is the median of residuals, and w_i is nonzero if $r_i < 6 \times median(|r|)$. The weighted least square is then implemented with robust weights and the local regression weights of equation (3.8) and all the new $\hat{t}$ are estimated and used to find residuals.

The weights are updated over p iterations until fitted values stabilize, where in practice $p \in [2, 5]$ works pretty well for most of the problems.

TABLE 3.2: Correct identification scores for the proposed features and the other methods after applying GlidarCo. LB and VB stand for the length-based and vector-based feature vectors, respectively. Features from all of the methods are computed from the corrected joints.

Method	**Average Accuracy(%)**	**Average F-score(%)**
[56]	43.40	38.43
[25]	28.33	26.25
[57]	73.01	73.83
[27]	74.52	71.68
GlidarCo, LB	73.66	69.60
GlidarCo, VB	<u>80.58</u>	<u>76.24</u>

3.2.1.1 Experimental results with GlidarCo

Table 3.2 summarizes the gait identification scores after employing joint correction using GlidarCo. By comparing the results with the gait identification scores before joint correction in chapter 2, we observe that GlidarCo can improve joint localization and identification scores in all of the methods. Besides, while our feature vectors in Table 3.2 do not contain the dynamics of the motion, vector-based features still outperform the methods that incorporate temporal information by computing moments of features over the gait cycle.

By comparing the results in Tables 3.2 and 3.1, we make some noteworthy observations. First, outlier removal and joint correction through GlidarCo both improve gait recognition scores. However, outlier removal also results in data elimination and loss of temporal information. Second, both methods acquire almost the same scores with vector-based features. But, compared with GlidarCo, outlier removal achieves higher scores with length-based features. One reason for such a result can be the elimination of data by outlier removal. This way, outlier removal is tested on much fewer data compared with other methods. Skeleton joint correction through GlidarCo can recover noisy and missing skeleton joints. However, joint correction is also prone to noisy estimation, in particular when there are missing joints over consecutive frames. On the other hand,

outlier removal also results in loss of temporal information. Later in section 3.2.3, we will show that by incorporating temporal information after joint correction, we can further improve classification scores.

3.2.2 Skeleton joint correction by GlidarPoly

In this section, we will describe how GlidarPoly acts on skeleton joint location measurements to correct and recover the noisy and missing skeleton joints. For each lidar video, given matrix L as was described in section 3.2.1, we perform filtering on L_m (each row of L), in two main steps: interpolation for the missing values, and robust smoothing to correct the outliers. Given the joint location sequence L_m, we find the sorted location of all the nonzero elements. We define n_{L_m} as the sorted set of all the indices in L_m with a non-zero value (each index corresponds with one time instant t_k) such that

$$\begin{cases} n_{L_m} = [n_1, n_2, ..., n_R] \\ n_1 < n_2 < ... < n_R \\ n_i \in [1, 2, ..., F_n]; i \in [1, 2, ..., R] \end{cases} \tag{3.10}$$

where R is the number of non-zero elements in L_m. Next, between any two nonzero values with non-consecutive indices along time, we fit a first-order polynomial through the least squares criterion

$$\begin{pmatrix} n_r & 1 \\ n_s & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} L_m(n_r) \\ L_m(n_s) \end{pmatrix} \tag{3.11}$$

where $n_r, n_s \in n_{L_m}$ and $n_s - n_r > 1$. $L_m(n_r)$ and $L_m(n_s)$ are the values of L_m at n_r and n_s, respectively. p_1 and p_2 can be obtained by finding the least squares solution to the system of equations in 3.11. Finally, we utilize RLOWESS (locally weighted scattered

plot smoothing) filter [67] to smooth the resulting joint location sequence and alleviate the effect of remaining lower-amplitude spikes in L_m.

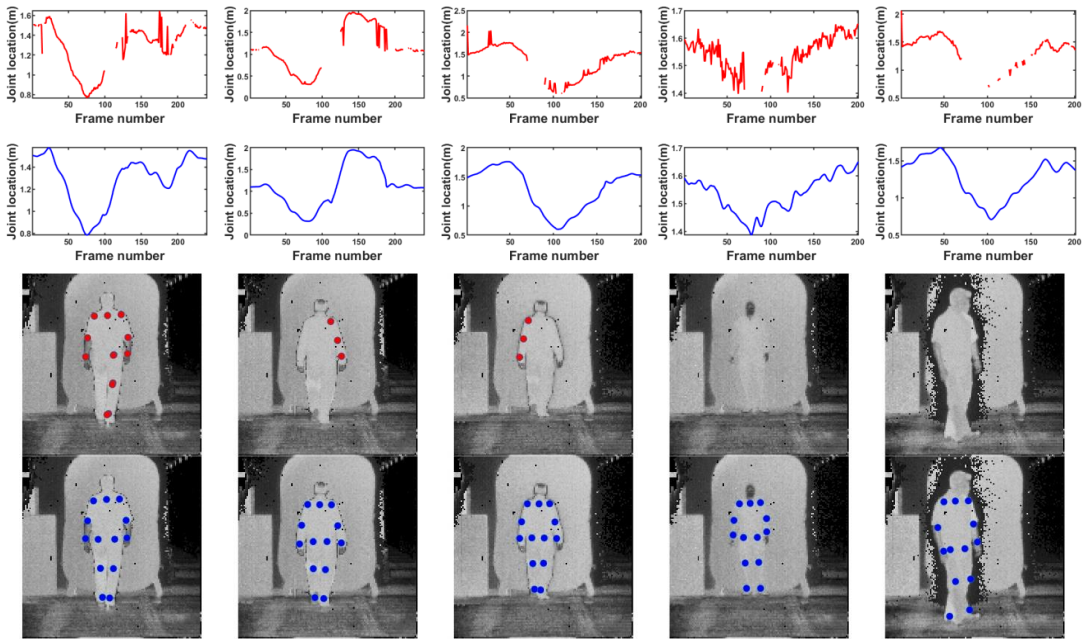


FIGURE 3.7: Effect of the skeleton joint location correction with GlidarPoly. From top: sample joint location sequences before (first row) and after (second row) joint location sequence filtering (each joint location sequence corresponds with one coordinate (x, y, z) of the location of one joint through time). Notice the abundance of missing values in the first row, which are shown as missing sections of the plotted signal that have been recovered through the joint correction (figures in the second row). The last two rows show samples of faulty and missing skeleton joints before (third row) and after (bottom row) joint location sequence filtering.

Figure 3.7 illustrates the result of applying GlidarPoly on samples of joint location coordinate time sequences and skeleton localization in the image reference frame. The results in this figure show the effectiveness of joint correction in interpolating and correcting missing and faulty joints. We observe that the original joint location sequences are noisy, containing many missing values and outliers. In the third row of Figure 3.7, we see the sample frames with missing skeleton joints in the image reference frame. As we observe in the last row, the missing joints are interpolated successfully through the filtering mechanism. We can also see examples where a whole skeleton is recovered through the joint location correction. The joint location correction can be easily applied in the cases of occlusion for the one-subject and multi-subjects scenarios. While in this study the missing joint locations are the result of erroneous joint localization, it can be the result of

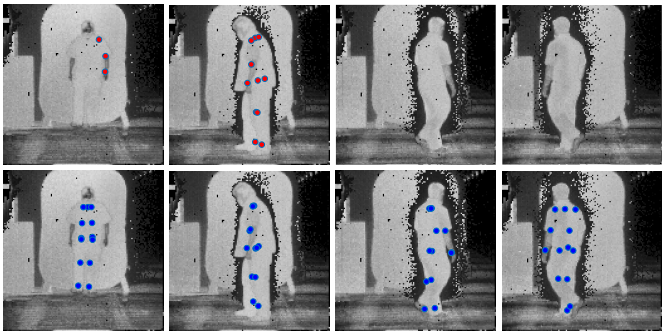


FIGURE 3.8: Failure examples of the joint location correction filtering. Sample frames of skeleton joints, before (top) and after (bottom) applying GlidarPoly for the skeleton joint location correction

occlusion. For most of the corrected skeletons, the interpolation of missing or noisy joints follows the correct joint locations. However, there exist cases where the obtained localization results are not accurate. Figure 3.8 shows a few failure examples of GlidarPoly in joint localization correction. The majority of such failure cases are the result of the existence of a considerable number of successive frames with missing or noisy joints that make the joint correction prone to faulty estimations. However, even for failure cases, at least half of the joints are predicted correctly. This can enhance the likelihood of correct identification compared to the original localization of the joints.

3.2.2.1 Computational complexity of GlidarPoly

The main computational bottleneck of GlidarPoly is in the last step, where we use RLOWESS [67] for smoothing the curve of joint location time sequences, and alleviate the effect of outliers with $O(Nlog(N)+3N(d+1)k)$ computational complexity. Here, N shows the number of points in a joint location time sequence, d is the degree of the polynomial used in the regression (here $d=1$), and k is the number of *k-nearest* point or length of each span in the local regression smoothing (k is constant and the same for all the points) [68].

TABLE 3.3: Correct identification scores for the proposed features and the other methods. LB and VB stand for the length-based and vector-based feature vectors, respectively. Features from all of the methods are computed from the joints that are corrected by GlidarPoly.

Method	**Average Accuracy(%)**	**Average F-score(%)**
[56]	40.77	36.21
[25]	32.55	32.49
[57]	80.11	80.40
[27]	75.79	72.75
GlidarPoly, LB	73.84	70.66
GlidarPoly, VB	**84.07**	**80.49**

3.2.2.2 Experimental results with GlidarPoly

Table 3.3 summarizes the gait identification scores after employing joint correction using GlidarPoly. By comparing the results with the gait identification scores before joint correction in Chapter 2, we observe that GlidarPoly can improve joint localization and identification scores in all of the methods. We do not consider the dynamic of the motion with any of the proposed features. However, our vector-based features still outperform other methods that incorporate the dynamics of the motion by computing the moments of features over the gait cycle.

By comparing results in Tables 3.2 and 3.3, we observe that both joint correction methods achieve almost the same improvement with length-based features. However, with vector-based features, GlidarPoly performs better. This observation can be due to over-smoothing the final correction estimation with GlidarCo that employs $3rd$-order polynomial in the first step of joint correction. By comparing the results in Tables 3.3 and 3.1, we also observe that GlidarPoly outperform outlier removal with vector-based features. With the length-based features, outlier removal acquires better results compared with GlidarPoly. For this observation, we can make the same argument as in subsection 3.2.1.1. Besides, since both joint correction methods achieve better performance with the vector-based features, we speculate that vector-based features are more robust to noisy estimations of joint correction compared with the length-based features.

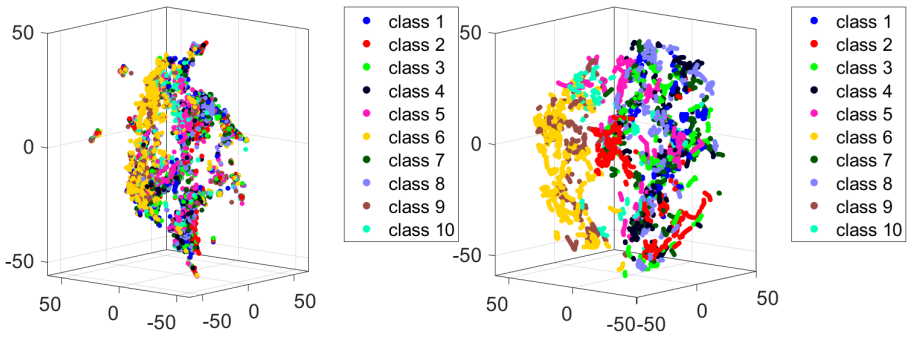


FIGURE 3.9: t-SNE visualization of the length-based features before (left) and after (right) applying the joint correction using GlidarPoly. There is a high level of inter-class intersection before joint correction (left) that is mostly resolved after correcting joint location, creating clusters that are more distinctive (right).

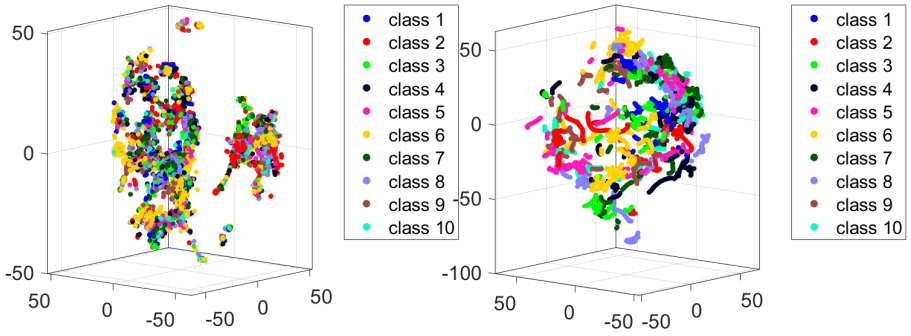


FIGURE 3.10: t-SNE visualization of the vector-based features before (left) and after (right) applying the joint correction using GlidarPoly. Before joint correction, high inter-class intersection and intra-class separation is observed (left). Joint correction transforms features into well-separated clusters (right).

Among the evaluated methods, we do not observe a considerable improvement in the performance of [25]. Authors in [25] use six angles between lower body joints as features and compute the mean, max, and standard deviation of each angle over every gait cycle. The skeleton model that we adopted in this work lacks foot joints that are essential to estimate two of the angles in [25]. To calculate these angles, we estimate the floor plane and use the normal vector to the plane. We speculate that the error in this estimation might also cause lower classification scores with this method. Besides, it was reported before that angle-based features might perform poorly compared with distance-based features, in particular when the number of subjects is relatively low [69]. Variations in the walking speed can also cause changes in the joint angles [70, 71].

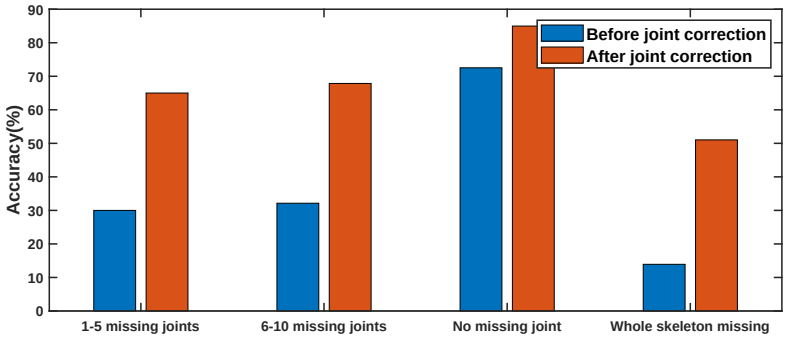


FIGURE 3.11: Comparison of classification accuracy for vector-based features based on the number of missing joints in the original skeletons, before and after applying GlidarPoly for joint correction. The samples with no missing joints also include noisy samples. All cases show improvement after applying the joint location correction.

In Figures 3.9 and 3.10, we present t-SNE visualization of length-based and vector-based features for the training data before and after joint location correction with GlidarPoly. Some of the interesting observations from these visualizations are as follows:

1. We observe a high level of the inter-class intersection before the joint correction for both features, that transforms into a wider separation among classes after applying GlidarPoly.

2. In the right graph of Figure 3.10, we see a non-homogeneous scatter of some of the classes, in particular class 9, which makes it more difficult to find the decision boundary. Such class distributions result in lower accuracy for these classes and overall lower accuracy for the whole dataset.

3. In the left graph of Figure 3.10, we observe two separate clusters that transform into a single one after joint correction (right graph).

4. After applying the joint correction, the transformed features become well separated, which shows we do not necessarily need a more sophisticated classifier.

Figure 3.11 presents the average identification accuracy before and after applying GlidarPoly, considering the number of missing joints in the originally detected skeletons. Samples with no missing joints also include noisy joint data. We observe that accuracy improves in all of the categories after the joint location correction, which confirms

the effectiveness of the joint correction in improving skeleton joint localization and gait identification.

3.2.3 Incorporating motion dynamics

As humans, we integrate both anatomy and the way that people move their bodies during activities such as walking, to recognize a familiar person. Features that describe motion play a crucial role in gait identification when different individuals have approximately the same body measurements. In several model-based methods, features like speed and step length are integrated to include the dynamic of the motion [56, 72]. Another common practice consists of computing mean, max, and standard deviation of selected features over every gait cycle and performing classification on such measurements [20, 27, 57]. This practice has proven to be successful in achieving high accuracy in gait recognition. However, the considered datasets generally have a low level of noise with a few to none outliers. Such datasets are recorded under controlled conditions such as limited walking directions.

The distance between two leg ankles, which is commonly utilized for the gait cycle estimation, has a cyclic pattern in general. However, variations in different walking factors such as walking direction, walking speed, and step length can cause aperiodicities in the walking patterns [20]. This can cause complexities in the interpretation of the motion, such as in gait cycle computation. In addition to such intra-personal variations in the gait, with the flash lidar data, there are numerous instances of consecutive frames with a missing skeleton. Therefore, the result of joint sequence correction is prone to noisy measurements. This noisy estimation will exacerbate the problem of the observed acyclic patterns. Figure 3.12 illustrates the ankle to ankle distance instances of flash lidar data. The sequence on the left shows a periodic pattern. However, the sequence on the right side of Figure 3.12 lacks a clear cyclic pattern. Irrespective of a sequence

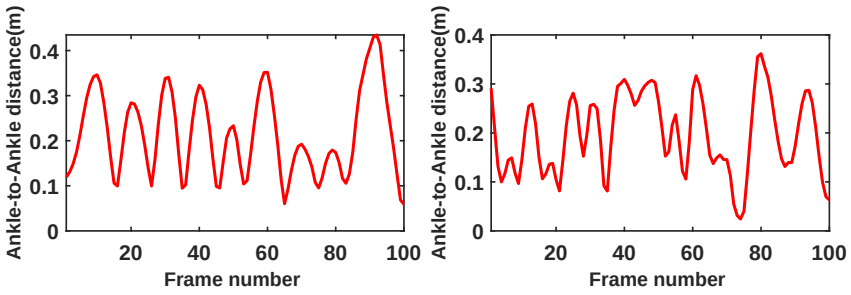


FIGURE 3.12: Two examples of the ankle to ankle distance sequence of lidar data after joint correction. While the graph on the left presents a periodic pattern, the sequence on the right lacks such a pattern.

being periodic or aperiodic, we consider a gait cycle as a local time sequence with three consecutive local maxima. To compensate for large variations in the gait cycle throughout one sequence of walking, we incorporate statistics that are robust to noisy measurements. In addition to commonly employed statistics of mean, standard deviation, and maximum, we also include median, upper, and lower quartiles that are robust to noisy data. We build feature vectors that comprise *mean, standard deviation, maximum, median, lower quartile, and upper quartile* of each feature over every gait cycle.

3.2.3.1 Results with motion dynamics

Table 3.4 shows gait recognition scores after joint correction when statistics of features are considered over gait cycles to incorporate the motion dynamics. For this experiment, we considered both joint correction methods with length-based and vector-based features. By comparing the results in Table 3.4 with the single-shot (per-frame) identification scores after joint location correction in Tables 3.2 and 3.3, we observe incorporating motion dynamics can improve identification scores for both features. We also observe that GlidarPoly acquires higher classification scores compared with GlidarCo. As we mentioned before, this observation can be due to over-smoothing of estimation with GlidarCo.

The average per-class accuracy and F-score for the single-shot (per-frame) case is presented in Table 3.5. We also show the per-class accuracy and F-score when statistics over the gait cycle are considered in Table 3.6. The results in both tables are computed

TABLE 3.4: Correct identification scores with statistics of features computed over gait cycle after joint correction. LB and VB stand for the length-based and vector-based feature vectors, respectively.

Method	**Average Accuracy(%)**	**Average F-score(%)**
GlidarCo (LB)	75.22	73.22
GlidarCo (VB)	84.65	80.38
GlidarPoly (LB)	76.03	74.88
GlidarPoly (VB)	**89.12**	**87.06**

TABLE 3.5: Correct identification scores for each class of subject for the single-shot scenario of vector-based features after applying GlidarPoly for the joint correction. The minimum and the next-to-lowest accuracy and F-score are underlined.

Subject #	**Accuracy(%)**	**F-score(%)**
subject 1	93.85	96.83
subject 2	80	79.69
subject 3	79.23	69.36
subject 4	<u>74.62</u>	<u>64.03</u>
subject 5	93.08	82.88
subject 6	76.92	<u>64.52</u>
subject 7	100	84.69
subject 8	76.92	85.29
subject 9	<u>66.92</u>	78.61
subject 10	82.31	88.25

TABLE 3.6: Correct identification scores for each class of subject for the statistics of vector-based features over the gait cycle after applying GlidarPoly for the joint correction. The minimum and the next-to-lowest accuracy and F-score are underlined.

Subject #	**Accuracy(%)**	**F-score(%)**
subject 1	<u>71.42</u>	83.33
subject 2	85.71	80
subject 3	85.71	92.31
subject 4	85.71	<u>75</u>
subject 5	100	93.33
subject 6	100	82.35
subject 7	100	<u>77.78</u>
subject 8	85.71	92.31
subject 9	85.71	92.31
subject 10	<u>78.57</u>	88

after applying GlidaPoly for the joint correction. By comparing the results in these two tables, we observe that the minimum per-class accuracy and F-score are improved by 4.5% and 10.97% as a result of employing gait cycle statistics. These results indicate that by employing features that capture motion dynamics, we can build a more reliable model compared to the case that only considers static features.

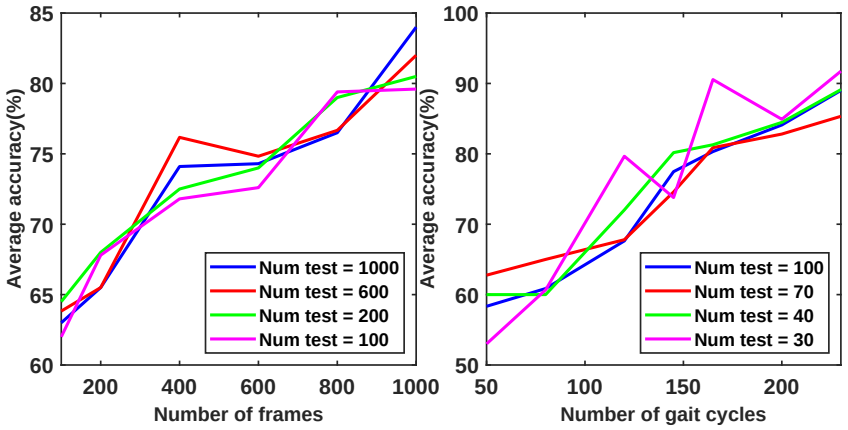


FIGURE 3.13: Average classification accuracy for different numbers of training samples given multiple numbers of test examples for the single-shot (left), and statistics over the gait cycle (right) scenarios. Both plots are acquired for vector-based features.

3.2.4 Effect of the number of training samples

In real-world applications, limited data availability is one of the main challenges of gait recognition. Therefore, it is essential to investigate how the designed model or the selected features perform under limited data availability. Here, we will examine the performance of the vector-based features, both for the single-shot scenario as well as the statistics over a gait cycle for different numbers of training observations given various numbers of test samples.

In Figure 3.13, the left graph illustrates the single-shot identification accuracy as a function of the number of training examples. For each experiment, we consider a different number of test samples, where the number of test samples changes in $[100, 1000]$ range. For a given number of test samples, the average accuracy improves as we increase the number of training data. For smaller number of test samples, accuracy increases at a higher rate when we use a larger number of training samples. A test sample size equal to or larger than 200 frames appears to be a proper choice empirically, as the accuracy trend shows to be more stable. We also observe that irrespective of the number of test data, we acquire the best performance with a training set of 1000 samples.

In the right graph of Figure 3.13, we show the average classification accuracy for different numbers of gait cycles for training. Each plot in this graph shows average accuracy for a certain number of test gait cycles. We observe that the highest accuracy is acquired with at least 200 gait cycles for training, irrespective of the number of test samples. When the number of gait cycles per subject is severely limited, this limitation can be problematic.

3.2.5 Evaluation on IAS-Lab

In this thesis, our focus is on the application of flash lidar modality for gait identification. However, due to the lack of publicly available flash lidar data for gait recognition, we evaluate the performance of the joint correction methodology on the *IAS-Lab RGB-ID* [73] dataset. For skeleton joints correction, we investigate the performance of GlidarPoly as it acquires higher classification scores. To evaluate the performance of joint location correction on *IAS-Lab*, we manually add noise to skeleton joints, as well as removing the whole skeletons in consecutive frames.

The *IAS-Lab* dataset includes the sequences collected from 11 different subjects. *IAS-Lab* consists of three sets, "Training", "TestingA", and "TestingB". The outfits of the subjects in "TestingA" are different from their outfits in the "Training" set. Sequences in "TestingB" are captured in a different room, but subjects wear the same outfits as in the "Training" set. In addition, some sequences in "TestingB" are recorded in a dark environment. In this experiment, first, we compute the single-shot rank-1 identification accuracy for the proposed vector-based and length-based features and compare it with several state-of-the-art methods. Next, we manually add noise to some of the skeleton joint locations and randomly remove some of the other joint location information. Then, we apply GlidarPoly and compare the results of gait recognition before and after applying GlidarPoly to correct the corrupted joints.

TABLE 3.7: Single-shot identification: Rank-1 identification accuracy for the proposed features, several RGB-based, and depth-based features for *IAS-Lab RGBD-ID* "TestingA" and "TestingB" sets. Dashes are for cases where no published information is available.

Method	**TestingA**	**TestingB**
RGB-based features		
HOG [74, 75]	31	47.21
Gabor-LBP [75, 76]	28.71	51.38
LOMO [75, 77]	26.37	30.97
Depth-based features		
Skeleton (SVM) [78]	—	—
Skeleton (NN) [78]	22.5	55.5
PCM+Skeleton [73]	25.6	63.3
3D CNN [79]	44.2	56.2
3D RAM [79]	48.3	63.7
ED+SKL [75]	48.75	58.65
Length-based (NN)	46.61	**70.64**
Length-based (SVM)	34.09	67.51
Vector-based (NN)	54.11	61.07
Vector-based (SVM)	**55.21**	67.71

Table 3.7 shows the single-shot rank-1 identification accuracy for the *IAS-Lab* dataset. We compare the performance of the proposed length-based and vector-based features with several RGB and depth-based methods. The results with the RGB-based features are reported according to [75]. As we can see, RGB-based features achieve better results on "TestingB" compared with "TestingA", where subjects are wearing different outfits. This is because changes in the outfit can affect the consistency of these types of features. Skeleton feature in [78] consists of 11 length-based features and 2 ratios of length-based features. PCM+Skeleton [73] consists of point cloud matching and the skeleton-based features of [78]. The 3D CNN [79] is trained on the 3D point cloud, and 3D RAM [79] is a recurrent attention model trained on 4D tensors of 3D point cloud over time. ED+SKL [75] is a depth-based feature, computed from eigen-depth and skeleton-based attributes. In the last four rows, we present the results with our length-based and vector-based features with both SVM and nearest neighbor (NN) classifiers. For the NN

classifier, we use the Manhattan distance with five nearest neighbors as in [80]. The results show that our vector-based feature outperforms other methods on "TestingA", where subjects are wearing outfits different from the training set. Besides, our length-based feature achieves the highest accuracy on "TestingB", where there are changes in the illumination.

3.2.5.1 GlidarPoly for joint correction in *IAS-Lab*

To evaluate the performance of the joint correction filtering on *IAS-Lab*, we added some error, using Gaussian distribution, to randomly-selected joints. We also randomly removed the joint location information of some other joints. Table 3.8 presents the single-shot rank-1 identification accuracy on *IAS-Lab* with the corrupted joints and after applying GlidarPoly for joint location correction. We observe that GlidarPoly improves the identification scores in the range of $[15\%, 33\%]$. Besides, we see that the identification accuracy after applying GlidarPoly is close to the results with the original data (the last four rows of Table 3.7). This observation confirms the effectiveness of the proposed joint correction filtering mechanism. Considering the length-based features in "TestingA", the results with GlidarPoly are even better than the results with the original uncorrupted data in Table 3.7. This suggests the removal of some of the noise that might exist in the original data. We also observe that improvement is more pronounced with the "TestingB" set. Considering the features, length-based features, in general, see a higher percentage of improvement after joint correction compared with the vector-based features.

3.2.5.2 Evaluation of gait cycle statistics

Table 3.9 shows the rank-1 identification scores after computing the six statistics of each feature over the gait cycles for the corrected skeletons. By comparing the results with the single-shot identification accuracy after joint correction in Table 3.8, we only observe improvements in three cases (shown in boldface). Earlier in subsection 3.2.4, we discussed

TABLE 3.8: Single-shot identification: Rank-1 identification accuracy for the proposed features on *IAS-Lab RGBD-ID* "TestingA" and "TestingB" before (with noisy and missing joints) and after joint location correction

Method	**TestingA**	**TestingB**
With added noise		
Length-based (NN)	23.86	30.18
Length-based (SVM)	31.20	41.43
Vector-based (NN)	28.05	31.76
Vector-based (SVM)	39.19	46.35
After applying GlidarPoly		
Length-based (NN)	48.24	63.93
Length-based (SVM)	48.09	63.01
Vector-based (NN)	52.44	59.57
Vector-based (SVM)	52.58	62.34

TABLE 3.9: Rank-1 identification accuracy using the 6 statistics of the proposed features on IAS-Lab "TestingA" and "TestingB" after joint location correction

Method	**TestingA**	**TestingB**
Length-based (NN)	**53.88**	**66.88**
Length-based (SVM)	45.88	**65.29**
Vector-based (NN)	50.24	53.56
Vector-based (SVM)	46.89	61.29

how our evaluation shows that we need an order of 10 gait cycles for training to acquire improvement over the single-shot scenario. With the lidar dataset, we required, on average, at least 20 gait cycles per subject to achieve such an improvement. In the lidar dataset, there is only one subject with less than 20 gait cycles for training. But, in the *IAS-Lab* dataset there are three subjects with such a condition. Therefore, we observe fewer cases of improvement in *IAS-Lab* compared with our flash lidar dataset.

In Figure 3.14, we show the performance of three sets of feature statistics over every gait cycle after applying GlidarPoly. The performance comparison is done on the lidar data, and "TestingA" and "TestingB" in *IAS-Lab*. We use NN and SVM as classifiers. In the majority of cases, the lower quartile, upper quartile, and median set outperform the mean, max, and standard deviation set after joint location correction. We also see cases where the former set can acquire higher identification accuracy compared with the

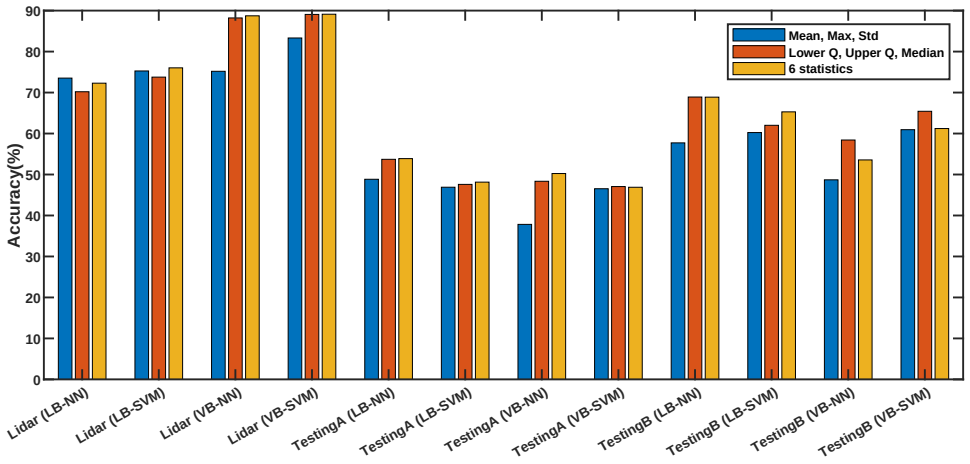


FIGURE 3.14: Comparison of the performance of mean, max, standard deviation set, and lower quartile, upper quartile, median set, and the set of all the six statistics to capture the dynamic of the motion after joint location correction. Comparison is performed for lidar and IAS-Lab datasets with both types of features and SVM and NN as classifiers. LB and VB stand for length-based and vector-based features, respectively. In the majority of cases, lower quartile, upper quartile, median set outperforms the mean, max, standard deviation set.

combination of all the six statistics. These observations suggest that lower and upper quartiles and median, as robust statistics, are better identifiers of temporal information when we employ joint correction to recover noisy and missing data. Our experiments also show that, in general, the combination of lower quartile, upper quartile, and median of the features in an SVM classifier framework yields about the same accuracy using either an RBF or a linear kernel. However, for the *IAS-Lab* dataset, the linear kernel performs better than the RBF kernel with the lower quartile, upper quartile, and median of vector-based features.

3.3 Remarks

In this chapter, we presented an efficient pipeline to improve the application of flash lidar for the gait recognition problem. The main challenge is caused by the low quality and noisy imaging process of flash lidar. Such signal quality adversely affects the performance of state-of-the-art algorithms for skeleton detection. The detected skeletons from the collected sequences contain a considerable number of erroneous joint location measurements.

Furthermore, the detections for several skeleton joints are missing in many frames. Under the described scenario, a common practice involves removing noisy data. However, data elimination results in the loss of temporal information and renders identification impossible in numerous frames, which is not desirable for time-critical applications, such as surveillance. To improve the quality of joint localization and to enhance gait recognition accuracy using flash lidar modality, we present methods for joint correction: GlidarCo, and GlidarPoly. We also present an automatic outlier detection method for applications where data elimination is not an issue. Furthermore, to incorporate motion dynamics after data correction, robust statistics are integrated that can effectively improve the performance of the designed features that only employ traditional feature moments over the gait cycles. The proposed pipeline is appealing in terms of computational complexity, scalability, and a simple, yet effective design.

Chapter 4

Gait anomaly recognition

Gait is a dynamic entity that changes over time. The relation between joints that are not directly connected can change from one pose to another (a.k.a. spatial variation). Besides, the relationship between joints that belong to different poses continuously varies through time (a.k.a. temporal variation). Among other objectives, the analysis of human gait can provide information that is beneficial for the medical assessment in the variety of physiological and neurological conditions [30, 31]. Researchers have evaluated gait patterns for diagnosis, progress assessment, and treatment evaluation of age-related impairments of locomotion, hip and knee osteoarthritis, post-stroke patients, multiple sclerosis, cerebral palsy, neurodegenerative disorders such amyotrophic lateral sclerosis, Huntington's disease, and Parkinson's disease [32, 33, 81].

Clinical assessment of gait is commonly conducted in specialized laboratory environments, using tools such as pressure-sensitive walkways, e.g., GaitRite, and marker-based systems, e.g., Vicon motion capture, that provide reliable gait data. However, such devices are costly and require extensive and elaborate sensor placement. Besides, such systems do not capture gait patterns that are observed in a natural environment [82], where the subjects show their habitual behavior. These patterns are vital for an accurate gait assessment. However, carefully supervised in-lab techniques are not a proper

representative of the gait patterns in a free-living environment. As life expectancy grows and with a growing elderly population, the development of appropriate frameworks for free-living gait assessment has become more vital. Such frameworks can be utilized for a frequent out-of-the-lab gait evaluation to detect early signs of age-related ailments that affect the patterns of gait. In recent years, numerous studies have attempted to address this issue by presenting frameworks for a free-living gait assessment [83–85]. Modalities such as wearable sensors and Kinect, which can collect spatiotemporal gait information out of specialized labs, have become the focus of numerous gait analysis studies for healthcare-related applications. The main advantage of Kinect is its low cost and markerless accurate human skeleton joint tracking capability that provides a convenient evaluation for the patients and the clinical laboratory technicians. Furthermore, multiple studies have investigated the liability of Kinect for clinical evaluation purposes [39, 40].

The majority of state-of-the-art gait anomaly detection methods have been focused on the evaluation of certain parameters of the gait. Examples of these parameters are gait speed, cadence (the rate at which a person walks, usually defined as the number of steps per minute), stride length (defined as twice the step length), and their alternation from the standard ranges in the healthy gaits. Numerous studies focused on gait features from certain lower limbs to recognize an abnormal gait. In [86,87], the authors performed a statistical analysis of selected gait factors. Daliri *et al.* [33] presented a time series analysis of stride intervals, swing intervals, stance, and double support intervals. Statistical analysis of the estimated probability density function of the stride signal was employed in [88] to detect abnormal gaits. In [89], authors used a wavelet-based characterization of stride time signals. An adaptive neuro-fuzzy inference on stride, stance, and double support intervals was presented in [90]. These studies are valuable, as they link the health evaluation of gait with relevant parameters that are clinically interpretable. However, they also limit the acquired information to a few selected factors and do not take into

account the interaction between different body segments in forming locomotion [91]. In recent years, success of the skeleton-based models in machine vision applications such as gait identification and activity recognition [52–54] has inspired numerous skeleton-based gait evaluation methods [3,92,93]. Such methods try to avoid the limitation in traditional approaches by taking into account the interactions between different body parts.

In this chapter, we employ the data collected via Kinect for a contact-free, marker-less approach to gait assessment. The skeleton data is captured by Kinect at a fixed interval. Subjects do not walk at the same speed throughout a whole recorded sequence. Besides, different subjects perform the same task at different paces that can create spatiotemporal patterns similar to other classes of normal/pathological gaits. These two scenarios can result in intra-class dissimilarity and inter-class similarities, which makes minor pathological gait recognition quite challenging. We present multi-class and computationally-efficient frameworks that can be adapted for convenient out-of-the-lab gait evaluation. First, following the recent surge of RNN-based networks in skeleton-based anomaly recognition, we present a multi-class gait anomaly classification framework that uses an LSTM network to detect embedded features in sequences of gait postures. Using the skeleton information provided by Kinect, we propose interpretable handcrafted features to represent each posture. For our first study, we will focus on a problem with large intra-class variations (a large collection of subjects) but a small number of class abnormalities. Next, we will consider a larger set of gait abnormalities, presenting a deep learning-based pipeline with minimal, yet effective skeleton data preprocessing. The presented framework can classify minor gait anomalies with high accuracy (state-of-the-art), can be applied to other datasets through transfer learning, and has the potential to be integrated into a free-living gait assessment framework for real-world applications.

4.1 Skeleton-based gait anomaly recognition (SGAR)

Skeleton-based **g**ait **a**nomaly **r**ecognition (SGAR) methods have become popular due to the low cost and data collection convenience of markerless modalities such as Kinect. Furthermore, skeleton-based features mimic real, interpretable human physiological traits. The majority of SGAR studies have been focused on handcrafted features and machine learning-based approaches. In this section, we briefly review some of the gait anomaly recognition methods that employ skeleton-based features and machine learning tools.

In [94], the authors employed the idea of joint motion history (JMH) to capture spatiotemporal motion information. Skeletons are normalized, a sliding window collects the normalized skeletons, which are transformed into a volume that is divided into voxels that capture the history of joint coordinates. The acquired JMH goes through dimensionality reduction, and abnormal gaits are detected by matching with a set of key pose templates. Meng *et al.*. [3] used the distance between 20 joints in the skeleton and employed a sliding window technique to capture temporal information. A random forest model was hired to detect abnormal gaits from the extracted spatiotemporal features. Paiement et al. [92] employed skeleton normalization for preprocessing and diffusion maps for dimensionality reduction. They built a statistical model of normal gait, and new observations were tested against the learned model of normal gait. In [2], the authors used a set of lower body limb angles to describe each pose. The resulting feature vectors were concatenated over every gait cycle, then transformed into a set of codewords using K-Means clustering. The normal gait model was built using Hidden Markov model, and a threshold based on the log-likelihood of such a model was used to recognize abnormal gaits. In [93], the authors build two covariance matrices, one for the mutual distance of joints in a skeleton, and another for the velocity of each joint. A custom covariance-based metric and K-nearest

classifier were used for abnormal gait detection. Khokhlova et al. [5] created a set of features based on low limb flexion angles and designed a long short-term memory (LSTM) network for classification. In [95], authors designed an autoencoder with recurrent layers for feature extraction and performed a series of classification experiments with multiple classifiers using the features from the encoder.

Among the proposed methods for SGAR, our work is closest to the approaches presented in [5], [95] and [4]. All of these studies employ deep learning-based techniques for the classification of an anomaly in a multi-class framework. A multi-class framework is unlike the traditional approaches in this field, where anomaly recognition is treated as either a one-class or a two-class problem. In [5], [95] and [4], an RNN-based network is designed for modeling and classification of gait abnormalities. Besides, the authors in [95] perform their experiments with subject cross-over between the training and test data. Finally, all of these studies evaluate their proposed model on one dataset only and have not demonstrated versatility for multiple scenarios.

Due to the dynamic nature of gait, sequential-based models such as Hidden Markov Model (HMM) and RNN-based networks such as LSTM and Gated Recurrent Units (GRU) have been the focus of many studies for modeling human gait [2, 5, 95–99]. In this chapter, we will focus on deep learning-based frameworks for a multi-class SGAR. To provide a fair evaluation, we design multiple deep learning networks for modeling and classification of spatiotemporal patterns of different normal/pathological gait categories. Thereby, before going into more detail about the proposed methodologies, we provide some background on deep learning and define major relevant concepts in this area.

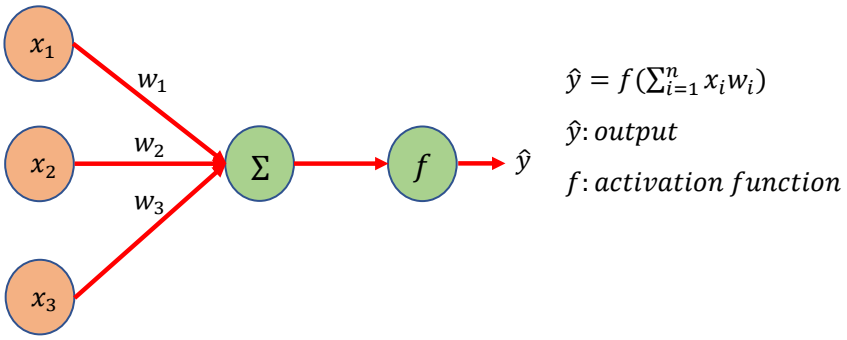


FIGURE 4.1: An illustration of a neuron (perceptron), the building blocks of a deep learning model. The activation function acts on the weighted sum of the inputs to create the output.

4.2 Some background on deep learning

Deep learning is a sub-field of machine learning, which itself is a sub-field of artificial intelligence. Artificial intelligence is a set of techniques that enables a machine to mimic intelligent human behavior. According to Tom Mitchell, a machine learning pioneer, in machine learning, the goal is to design computer algorithms that can improve automatically through experience. Traditional machine learning algorithms require hand-engineered features for prediction and classification. Inspired by the structure of the human brain, deep learning models extract underlying features directly from data to carry on a certain task. After the surge of modern deep learning models in 2006 [100], deep learning has revolutionized various fields from computer vision and robotics to natural language processing, finance, and medicine.

Deep learning models consist of multiple layers of neurons (perceptrons). Figure 4.1 shows a neuron. Each of the x_is in this figure represents one input that has its weight of w_i. The weighted summation of the input is calculated as is described in this figure and passed to an activation function that produces the output. In general, each neuron also has a bias term that is added to the other weighted inputs and shifts the activation function to the left or right. We can write the equation in Figure 4.1 in a vectorized

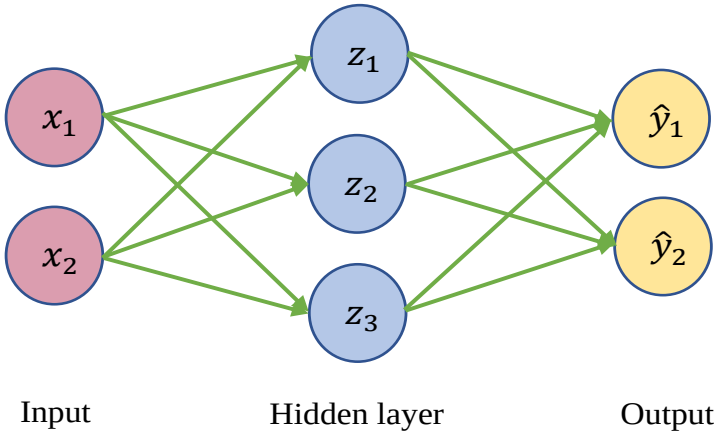


FIGURE 4.2: A simple neural neural network that consists of one dense hidden layer. The hidden layer is called a dense or fully-connected layer because all the inputs are fully/densely connected to all the outputs.

format. By adding the bias term, the output of a neuron can be written as follows

$$\hat{y} = f(\mathbf{b} + X^T W) \tag{4.1}$$

where f is the activation function, $\mathbf{b}$ is a vector of biases, X^T is the vector of inputs, W is the matrix of weights, and $\hat{y}$ is the vector of outputs. For a multi-layer network, this equation describes the relationship between the inputs and outputs of each layer. A neural network has at least one hidden layer. A simple neural network is shown in Figure 4.2 with two inputs, two outputs, and a single hidden layer that consists of three neurons. This figure is an example of a dense or fully-connected network since all the inputs are densely connected with all the outputs.

The outputs of each layer of a neural network are the multiplication of its inputs (or the output of the previous layer) and its weights. Thereby, each layer of a neural network is a linear function on its own. A neural network requires non-linearity for modeling highly complex and nonlinear functions, and ***activation functions*** are responsible for introducing non-linearity to the model. Thereby, activation functions are generally nonlinear functions. There are various types of activation functions. For a long time, sigmoid function has been the most commonly used activation function. However, sigmoid

outputs the same value for a large range of positive and negative inputs (+1 and −1, respectively). This makes training difficult, as the activation function can saturate and the model stops to learn after some time. In recent years, rectified linear unit or ReLU has become the most widely used activation function. RELU is the default activation function in dense and convolutional layers. RELU is partly inspired by a neurological analogy that the neurons in our brain are either inactive or activated. The ReLU function is a piece-wise linear function that will produce an output of zero for a negative input and outputs the same input otherwise

$$ReLU(x) = max(0, x) \tag{4.2}$$

Therefore, if the input of a neuron is beyond some *threshold*, the neuron acts like an identity function. ReLU and other piece-wise activation functions preserve many properties of linear functions such as easy optimization by gradient-based algorithms. Thereby, training with ReLU is fast and fairly easy.

To approximate a proper function, a deep neural network needs to learn the right weights and biases. This is done through a learning phase, in which a deep model approximates a function to predict or classify, such that the cost of incorrect predictions is minimized. For each input, a loss or cost function tells the difference between the actual value or label of that input and the prediction of the model for that input. An ***empirical loss*** can be written as the average of losses over each training example

$$\mathcal{J}(W) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i; W), y_i) \tag{4.3}$$

where W is the weight matrix, x_i shows the input for training example i, and y_i shows the true value or label of input example x_i. The goal is to find W^*, the set of values for weight matrix W, such that $W^* = \arg\min_W \mathcal{J}(W)$. There are different types of

loss functions, such as mean square error (regression), hinge loss, cross-entropy, and KL-divergence (classification). We use the latter two loss functions in our problem and provide more detail about them later in this chapter.

Once a loss function is formulated, the weights of a neural network are initialized randomly. The goal is to find a set of weights that minimize the loss function. These weights are determined through an iterative procedure until some conditions are satisfied. In each iteration, gradients of the loss function with respect to all the weights (and biases) are computed, $\frac{\partial \mathcal{J}(W)}{\partial W}$. The gradient of the loss function with respect to each weight tells us how sensitive the loss function is to the change in that weight. Each gradient has a value, which shows how important each weight is, as well as a direction, which tells us in what direction the cost should move. The algorithm that is used for computing the gradient of the loss function with respect to each of the weights is called ***backpropagation***. Backpropagation uses the chain rule to compute the gradients. The gradients are used for updates to the weights according to the following

$$W \mapsto W - \eta \frac{\partial \mathcal{J}}{\partial W} \tag{4.4}$$

where η represents ***learning rate*** that determines the step size in the direction of the gradient, toward the minimum of the loss function at each iteration [1]. Computing the gradients of the loss function and weight update according to Equation 4.4 is done in each iteration until a local minimum is reached. This optimization algorithm is known as ***gradient descent*** and is widely employed in finding optimum parameters in deep learning models. Apart from gradient descent and its derivatives (stochastic gradient descent and mini-batch gradient descent), deep learning has seen a surge of numerous optimizers over the last years. Many of these optimizers, such as Adagrad, Adadelta,

[1]Biases are updated according to an equation that looks the same as Equation 4.4, the only difference is that gradient of loss with respect to bias is used instead: $b \mapsto b - \eta \frac{\partial \mathcal{J}}{\partial b}$. From now on, everywhere you see a weight update equation, consider the same equation for bias update by substituting $\frac{\partial \mathcal{J}}{\partial W}$ with $\frac{\partial \mathcal{J}}{\partial b}$.

RMSProp, and Adam are based on adaptive learning-based approaches. Later, we will describe each of the optimizers that are utilized in designing any of the deep learning models in this chapter.

4.2.1 Overfitting

One of the problems that happen during the training phase of a deep learning model is overfitting. Overfitting occurs when the training loss of the network keeps improving and reaching a very small value, but when we evaluate the network on unseen data, it shows a large loss. This occurs because the network is overly trained on the training data. At this point, the network stops learning features that could be generalized and only memorizes the training data.

Deep learning models are notorious for their large number of parameters. The complexity of deep networks enables them to model complex functions. The high complexity can also result in overfitting because deep models learn the patterns from data. More parameters in a network require more data for training. In particular, this can be problematic when the function that the network is modeling is too complex. One way to prevent overfitting is to reduce the complexity of the model. We can reduce complexity by reducing the number of parameters, such as the number of layers or the number of neurons in each layer. Adding a penalty to the loss function is another way to reduce the complexity of a deep model.

With more data, models generally improve their performance. However, after some point, the performance of classic machine learning models plateaus, while a deep learning model keeps improving until it reaches its capacity of learning. More complex models have a higher capacity of learning and require more data to learn generalizable patterns, or they might overfit otherwise. Thereby, adding more data or creating more data through data augmentation is another way to reduce overfitting in a deep learning morel.

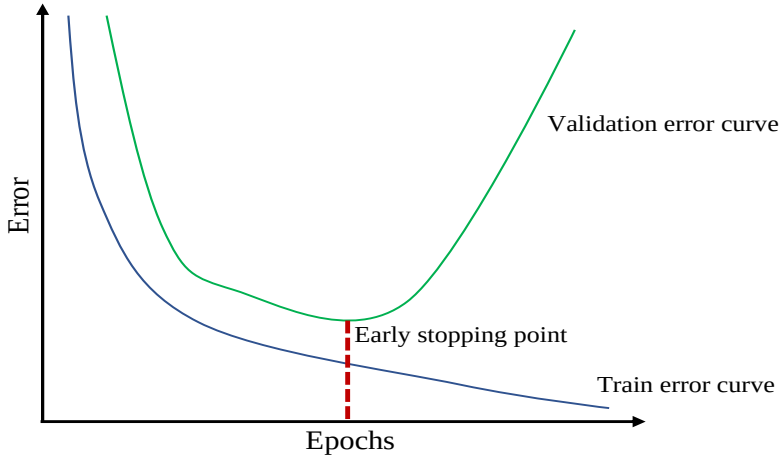


FIGURE 4.3: Illustration of early stopping. The validation error starts increasing after some point, while the training error keeps decreasing. With early stopping, model parameters at the early stopping epoch are saved for evaluation on the test data.

There are other regularization techniques, such as dropout, early stopping, weight decay, and activity regularization for reducing overfitting in a deep model. In the following subsections, we will describe some of these methods that have been employed in our models' design.

4.2.1.1 Dropout to prevent overfitting

Dropout [101] works as a regularization technique in deep neural networks and helps to prevent overfitting and reduce the generalization error. On each training iteration, each node of the network, along with all its in-going and out-going connections, will be eliminated with some probability. At each iteration, this process will generate a smaller network, preventing each node from overreliance on the nodes that are connected to it. In practice, dropout will result in a structure with smaller weights, acting like an adaptive $\mathcal{L}2$ regularization [102].

4.2.1.2 Early stopping to prevent overfitting

Early stopping [103] is another regularization technique to avoid overfitting. Figure 4.3 provides an illustration of early stopping. As we observe in this figure, as the training

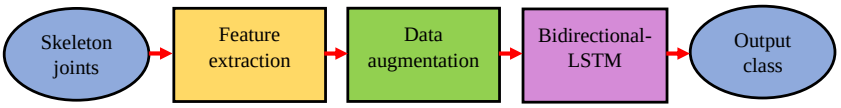


FIGURE 4.4: Pipeline for gait anomaly recognition with handcrafted features and bidirectional LSTM

epoch [2] increases, the training error decreases. However, by looking at the validation error [3], we see that the error decreases to some point and then start increasing. This behavior is a sign of overfitting. The network is just memorizing the training data after the early stopping point, thereby cannot be generalized to the validation data. In early stopping, the training stops at the early stopping epoch based on a criterion [4], and the parameters of the model are saved for later evaluation on test data. Early stopping has a hyperparameter that is called *patience*. Patience is the number of epochs that the training continues after the early stopping epoch. The validation error curve is not always as simple as the one in Figure 4.3. Sometimes, the validation starts increasing and after some point starts decreasing again and reaches a new minimum, and sometimes this increase and decrease in the validation error curve happens more than once. We use patience for such scenarios because we always want to save the model that reaches its high performance on the validation data.

4.3 SGAR with handcrafted features

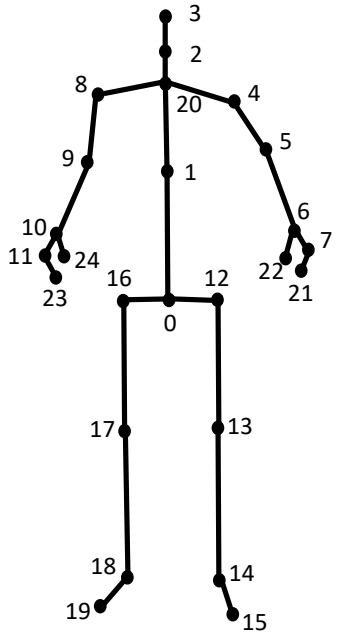
Figure 4.4 describes the pipeline for the gait anomaly recognition using handcrafted features from the skeleton data. The inputs to the presented SGAR system are sequences of skeleton joints $J = [J_1, J_2, ..., J_f]$, where f is the number of frames in a sequence and

$$J_i = [x_k, y_k, z_k]_{k=1}^{25} \in \Re^{3N} \tag{4.5}$$

[2] An epoch is the single pass of all the training data in a neural network.

[3] The network does not use validation data for training. Validation data is used for model evaluation during the training phase.

[4] like the epoch at which validation error reaches minimum or validation accuracy reaches its maximum.



0: Spine base
1: Mid-Spine
2: Neck
3: Head
4: Left shoulder
5: Left elbow
6: Left wrist
7: Left hand
8: Right shoulder
9: Right elbow
10: Right wrist
11: Right hand
12: Left hip
13: Left knee
14: Left ankle
15: Left foot
16: Right hip
17: Right knee
18: Right ankle
19: Right foot
20: Spine shoulder
21: Left hand tip
22: Left thump
23: Right hand tip
24: Right thumb

FIGURE 4.5: Skeleton joints for Kinect v2

with (x_k, y_k, z_k) representing the kth joint coordinates in frame i. We use skeleton joints generated by a Kinect camera as the input. Figure 4.5 shows joint information that is generated by a Kinect camera. Features in each frame are extracted from skeleton joints. In the next step, data augmentation is performed on the designed features. Finally, the output of the augmented features is fed into a bidirectional LSTM network for modeling the gait patterns and classification. In the following subsection, we describe each building block of the presented pipeline in more detail.

4.3.1 Feature extraction

Previous studies have shown that lower-body limbs can provide reliable information for gait representation and detecting anomalous gaits for various types of gait disorders [2,5,104–106]. Following this direction, we propose a set of attributes based on lower body joints. Figure 4.6 shows the proposed feature vector. As we can see in this figure, the proposed feature vector consists of six 3-dimensional (3D) vectors and four angles [2, 25]. Therefore, each frame is represented by a concatenated vector of these ten features,

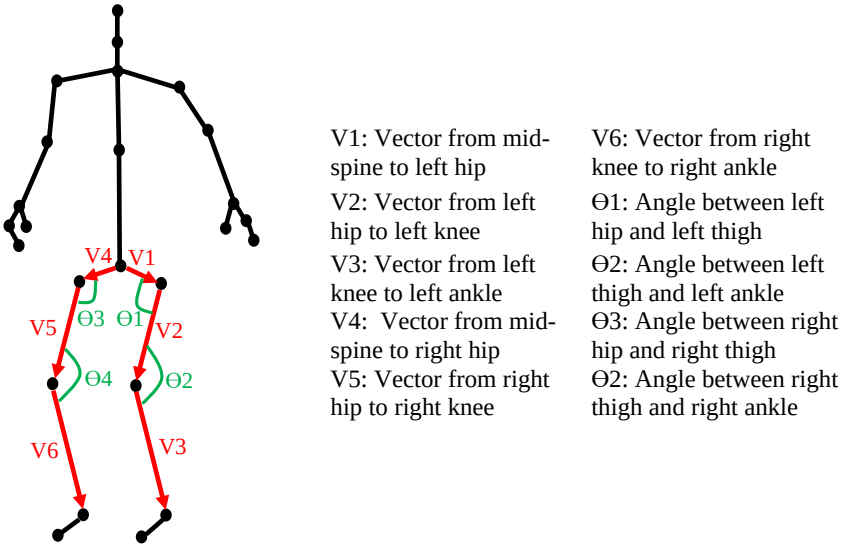


FIGURE 4.6: Representation of the handcrafted features for skeleton-based gait anomaly recognition. There are six 3-dimensional (3D) vectors that are shown in red and 4 angles that are depicted in green.

creating a feature vector of 22 dimensions. Each of the 3-dimensional vectors presents a limb vector in the lower body and is the same as part of the vector-based feature vector that we designed for gait recognition with flash lidar data (refer to Chapter 2.1.1) [80]. Previous studies have shown that the joints in the foot are prone to noisy measurements [107, 108], and our anecdotal experiments support this conclusion. Therefore, we do not consider the angles and 3-dimensional vectors in the foot area.

4.3.2 Data augmentation

The main superiority of deep learning comes from its ability to learn nonlinear patterns in a high-dimensional space. However, to design deep models that can extract such high-level features and can generalize well to unseen data, we need a large number of data for training. While acquiring a considerable number of data might not be an issue for applications in computer vision and natural language processing, for some tasks, such as gait anomaly recognition, it is challenging. This difficulty is mainly due to the high cost of data collection. Most of the skeleton-based datasets for anomaly classification are not publicly available due to confidentiality concerns for patients. There are only a

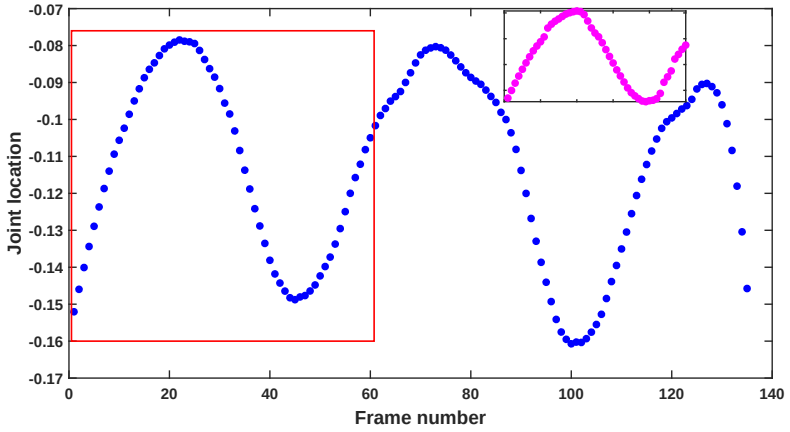


FIGURE 4.7: Illustration of data augmentation using window warping for a sample joint coordinate sequence. The small plot on the top right of the figure shows time-ordered samples of the values in the red window that have been selected uniformly at random.

few publicly available datasets that do not contain a large collection of observations and include anomalies that are simulated by healthy subjects. Under such conditions, data augmentation can be a valuable tool for improving the performance of a deep model.

In computer vision, multiple data augmentation techniques such as rescaling, flipping, cropping, and rotating have been purposed and successfully employed to improve the performance of a model in a variety of different tasks. Unlike computer vision studies, the application of augmentation has been limited in time series-based problems with deep learning. Many augmentation methods in computer vision cannot be directly employed on time-series datasets. Furthermore, with time-series data, data augmentation is not trivial, as it heavily depends on the nature of the dataset and the context of the problem. We can look at SGAR as a multivariate time-series classification problem, where each time sequence represents joint coordinates or a set of feature values through time. However, due to the complex relationships between different features or skeleton joints during motion, data augmentation is not trivial for such type of problems. Besides, for datasets with high inter-class similarity, data augmentation becomes even more challenging.

In this study, we use sequences of features that are computed using the lower-body joints coordinates. Here, we perform data augmentation in two steps. First, we divide all

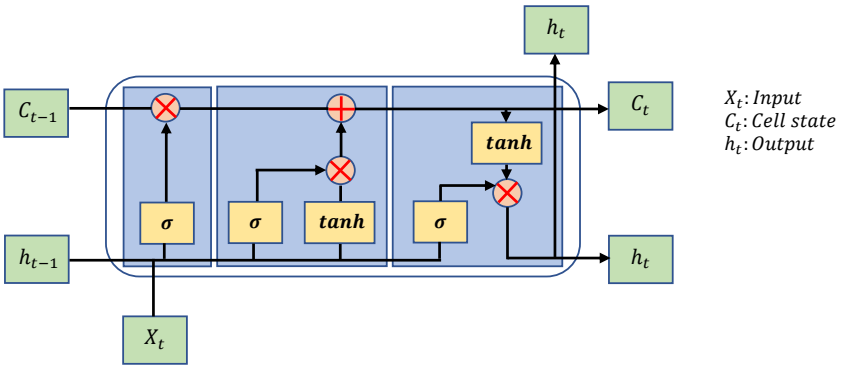


FIGURE 4.8: Structure of an LSTM cell. Each block with σ shows a sigmoid function. By creating a value between 0 and 1, sigmoid functions act as the gating function, controlling the flow of information. h_t and C_t are hidden and cell states that are passed to the next LSTM cell that also takes X_{t+1} (input at time step $t+1$) as the input. The building blocks of an LSTM are shown by three blue blocks inside the LSTM cell. From Left to right, these blocks are: forget gate, input gate, and output gate.

the available sequences into sequences of smaller lengths, where all the resulting sequences will have the same length of L (there are L frames in each resulting sequence) [109]. The next step of data augmentation involves window warping [110]. From the original sequences, we select sequences of the same length of M where $M > L$. Then we down-sample uniformly to remove extra frames and generate sequences of the same length of L. Figure 4.7 illustrates window warping for a sample of joint coordinate sequence. We can determine L based on the condition under which the data was collected. For example, we can use the reliable range of Kinect camera and the average number of frames that can be collected when a subject is walking in front of the camera in that range. In general, both L and M can be determined by experimentation. Next, we will provide some background on LSTM networks and describe the architecture of the designed LSTM model.

4.3.3 Bidirectional LSTM

In this subsection, we will describe the structure of the designed bidirectional LSTM for modeling and classifying the gait patterns. But, before going into more detail about the network architecture, we will provide a brief background on LSTMs. Also, we explain the

stochastic weight averaging that is used in the design of the network to reduce variance and generalization error.

4.3.3.1 Some backgrounds on LSTM

LSTM is a specific type of recurrent neural network (RNN) with memory blocks and a gated structure. While RNNs can work with problems that involve short-term dependencies, they fail to capture the context for problems with longer sequences. A neural network uses gradients of a cost function to update the weights and biases in the network such that the desired cost function of the network is minimized. The gradients of the cost function are computed using the backpropagation algorithm [111]. In RNNs, the gradient solution backpropagates through time [112]. As the gap between relevant information in a sequence increases, the gradient shrinks that disables the network from learning. The shrinkage of gradients during backpropagation is called the vanishing gradient problem [113] and is the reason behind the failure of RNNs with long-term dependencies. The LSTM [114] was introduced to address the problem of RNNs with long sequences. The gated structure of LSTMs regulates the flow of information, helping the network to forget the unnecessary information and retain what is essential.

Figure 4.8 shows the structure of one LSTM cell. Like any RNN, each LSTM layer is a chain of repeating LSTM cells, where the cell state and output at time step t will be fed to the next LSTM cell at the time step of $t+1$. Each LSTM cell has three gates that are shown by three blue blocks in Figure 4.8. Given $x_1, ..., x_{t-1}, x_t, x_{t+1}, ..., x_T$ that is a sequence of length T, first it is decided what information to forget through a sigmoid function in the forget gate

$$\begin{aligned} f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\ \sigma(p) &= \frac{1}{a + \exp(-p)} \end{aligned} \tag{4.6}$$

where U^f and W^f are the weight matrices of the forget gate. The input gate decides which information to update

$$i_t = \sigma(x_t U^i + h_{t-1} W^i) \tag{4.7}$$

with U^i and W^i defining the weight matrices of the input gate. Finally, the output gate provides activation to the LSTM cell at time step $t + 1$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \tag{4.8}$$

where U^o and W^o are the weight matrices of the output gate. The candidate cell state $\tilde{C}_t$, the cell state C_t, and the output h_t can be described by the following formulas

$$\begin{aligned} \tilde{C}_t &= \tanh(x_t U^g + h_{t-1} W^g) \\ C_t &= \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \\ h_t &= \tanh(C_t) * o_t \end{aligned} \tag{4.9}$$

where $*$ shows a point-wise operation. In an LSTM cell, sigmoid function (σ) is called the recurrent activation and tangent hyperbolic, $tanh(p) = (1 - \exp(-2p))/(1 + \exp(-2p))$, is called activation. The role of an activation function is to introduce non-linearity to the networks that is required for modelling complex non-linear processes.

LSTMs can detect patterns in sequence-based data with their gated structure that removes the unnecessary information and retains the relevant context. The bidirectional LSTM is a specific variety of LSTM networks in which each sequence is presented in both forward and backward order. Unlike unidirectional LSTMs that only look at the past context, a bidirectional LSTM can utilize both past and future contexts [115]. Bidirectional LSTMs have shown superior performance compared to their unidirectional counterparts in a variety of sequence-based problems [116–118].

4.3.3.2 Stochastic weight averaging

The loss landscape of many deep learning models contains various sharp minima. These minima are often caused by a combination of noisy/anomalous training data and the structure of the model. As such, small changes in the input can result in a very different solution, causing high variance. With smaller datasets, the effect of changes in the input on the prediction of the model can be more dramatic. In a small dataset, the variance in the input can be high, but the lack of enough data avoids the network to discover the hidden relationships between input and outputs.

To achieve low generalization error, it is desirable to use an algorithm that can avoid sharp minima [119]. While prevalent gradient-based optimization algorithms for training deep learning models are flexible in exploring the solution space, they are not capable of avoiding/detecting such sharp minima. Designing an ensemble model is a solution to mitigate the effect of such minima, which ultimately reduces the variance of a deep learning model. Besides, ensemble models can also improve the performance of the model, resulting in higher accuracy. However, training a deep learning ensemble is time-consuming and memory-demanding. Stochastic weight averaging (SWA) [120] offers a solution to the computational inefficiency of deep learning ensembles.

SWA takes multiple snapshots of the model during the training phase and computes an average over the weights of the model in each of the saved snapshots. The aggregated weight average is updated according to the following equation:

$$w_{SWA} \leftarrow \frac{w_{SWA} \times n_{models} + w}{n_{models} + 1} \tag{4.10}$$

where w is the network weights at the epoch that SWA is initialized, and n_{models} is the number of models at the end of each epoch in which w_{SWA} is updated. If c defines the length of the cycle for a cyclic learning rate ($c = 1$ for constant learning rate), w_{SWA} gets an update at the end of epoch for which $mod(i, c) = 0$, where i is the epoch number.

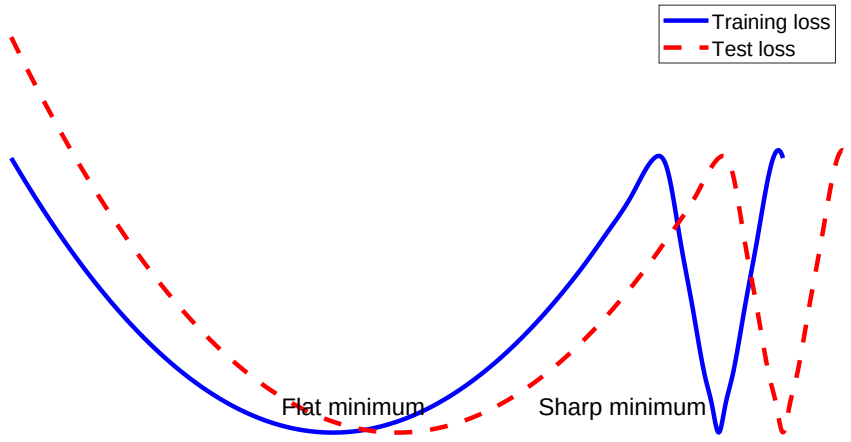


FIGURE 4.9: Conceptual visualization of Flat and Sharp Minima. The Y-axis represents values of the loss function and the X-axis shows the variables (parameters) [1]. If the learning algorithm lands on a solution in the weight space that corresponds with a sharp minimum in the training loss surface, the test loss at the same point might be a large value. On the other hand, finding a solution in the flat region of training loss will most likely result in a small value for the test loss function.

The goal of a deep learning model is to find a point in the high-dimensional weight space such that the loss function acquires a low value both on training and test set. To achieve this goal, the learning algorithm of a deep learning model travels the weight space during the training phase. The main idea is that train and test loss surfaces are similar but not the same. Imagine the learning algorithm has found a solution in the weight space that corresponds with a sharp minimum of the training loss surface. Then, for a slightly shifted test loss surface, this solution might result in a big value for the test loss. By averaging in the weight space, SWA leads stochastic gradient descent (SGD) toward wide flat regions of the loss surface. So, a point that results in a low value in the training loss surface also acquires a low value in the test loss surface, which corresponds with a lower generalization error. Figure 4.9 shows simplified loss surfaces in 2D for a sharp and flat minimum scenarios. By taking the average in the weight space, SWA offers the benefits of an ensemble, such as low variance and higher accuracy, without the computational overhead of traditional ensemble models that happen in the model space [120]. Previous studies have demonstrated performance improvement with SWA in different fields such as computer vision [120], language modelling [121], semi-supervised

TABLE 4.1: Description of the bidirectional LSTM model for gait anomaly recognition. Number of units shows the number of units in each LSTM layer.

Hyperparameter	Value	Hyperparameter	Value
Number of layers	2	SWA learning rate	0.01
Number of units	256	Epochs	110
Batch size	16	Optimizer	SGD
Learning rate	0.1	Loss function	Cross-entropy

learning [122], and Bayesian model averaging [123]. We implement SWA in the design of the proposed bidirectional LSTM model to reduce the generalization error of the model in predicting the class of anomalous gaits.

4.3.3.3 Description of the bidirectional LSTM architecture

Table 4.1 shows a description of the designed bidirectional model for gait anomaly recognition. The model is trained with mini-batches of 16 samples. The inputs to the network are all scaled in the $(-1, 1)$ range. Rescaling all the features to a certain range is essential because each feature vector consists of angles and 3-dimensional vectors that have different scales. For each feature $v_i \in [r_{min}, r_{max}]$, where r_{min} and r_{max} represent the lower and upper limit of the range of values that feature v_i takes over all the training examples. In general with $[t_{min}, t_{max}]$ as the target range, each feature like v_i is rescaled according to

$$v_i \mapsto \frac{v_i - r_{min}}{r_{max} - r_{min}} \times (t_{max} - t_{min}) + t_{min} \tag{4.11}$$

Here, $[-1, 1]$ is the target range. Studies have shown that feature scaling can lead to faster convergence and also avoids the network becoming stuck in local optima [124]. Each mini-batch consists of sequences of the features that are drawn from lower body joints (refer to subsection 4.3.1). Each sequence has a dimension of $R^{35\times22}$, with a sequence length of $L = 35$, and 22 features per time step.

Each bidirectional LSTM layer has 256 neurons with a dropout of 50%. The network is trained for 110 epochs, and SWA is initialized roughly in the last half of the epochs ($epoch = 61$). To compensate for the class imbalance in this dataset, we use weighted

categorical cross-entropy as the loss function

$$J = -\frac{1}{M}\sum_{k=1}^{K}\sum_{m=1}^{M} w_k y_m^k \log(h_{\theta(x_m,k)}) \tag{4.12}$$

where M is the number of training samples, K is the number of classes, y_m^k is the k'th element of one-hot-encoded target label of example m, h_θ is the k'th element of the prediction of neural network for example m, and w_k is the weight for class k. Here, we define class weights according to the following formulation

$$w_i = \frac{max(\bigcup_{k=1}^{K} N_k)}{N_i} \tag{4.13}$$

where w_i is the class weight for class i, N_i is the number of samples in class i, and K is the number of classes in the dataset. As we observe, the weights of each class depends on the population of that class, as well as the population of the majority class. This way, the least populated class gets the highest weight and the majority class gets the unit weight. Thereby, to avoid a bias of the loss function toward more populated classes, we penalize the miss-classification in classes with less samples with larger weights. To optimize the loss function, we use *mini-batch gradient descent*

$$\theta^{new} = \theta^{old} - \eta\nabla_\theta J(\theta; x^{(i:i+n_b)}; y^{(i:i+n_b)}) \tag{4.14}$$

where θ is the model's parameters, η is the learning rate that determines the step-size toward a (local) minimum at each iteration of the model's parameters update, $\nabla_\theta J$ is the gradient of the loss function with respect to its parameters, and n_b is the batch size. Mini-batch gradient descent is a variation of gradient descent, where a subset of training data is used to update the weights. Compared with batch gradient descent that uses all the training data for each weight update, mini-batch gradient descent is computationally

efficient; however, it adds another hyperparameter to the model, which is the batch size.

In this dissertation, every time we employ the SGD (stochastic gradient descent) as the optimizer, we are using mini-batch gradient descent with learning rate decay and momentum. *Learning rate decay* is defined according to the following equation

$$lr = lr_{in} * \frac{1}{1 + iteration * decay} \tag{4.15}$$

where lr_{in} is the initial learning rate and decay is the amount of weight decay at each iteration. With learning decay, the learning rate is updated according to Equation 4.15 at the end of each epoch. Using learning rate decay can help in converging closer to the minimum of the loss function compared with a fixed learning rate. A learning rate defines how strongly we move in the direction of gradient descent at each mini-batch update. Generally, we might afford to take larger steps in the beginning to accelerate training. But, as we get closer to the minimum of the loss function, taking large steps might cause overshoot or wander around the solution. On the other hand, if the learning rate decays over time, as we get closer to the solution, the steps toward the solution become smaller. This will land the model in the small neighborhood of the minimum (an area of lower loss), helping the convergence of the model toward an acceptable accuracy [125].

Another parameter in stochastic gradient descent optimizer is *momentum* [111]. Momentum can be helpful in two ways: it can accelerate convergence and can lead to a smoother convergence. Mini-batch gradient descent and gradient descent [5] provide a noisy weight update, since each time, we only use a subset of training data for weight update. As a result, weight updates cause oscillations, and this will slow down the convergence. Besides, the training loss function has a complex landscape with many narrow

[5] In gradient descent, parameters are updated with every training example

valleys and local minima, where loss varies differently in different directions [6]. To get a solution that can be generalized well to unseen data, we need to avoid shallow local minima. We also need to reach a flat minimum, such that a shift in the loss surface of unseen data does not land the model on a large loss value [7]. As gradient descent leads the model toward a proper local minimum, the model might get trapped in shallow or narrow valleys of the loss landscape as gradient values are not large enough to push the model out of such local minima. Momentum adds a boosting term to the gradient descent by taking an average over several (or few depending on the momentum coefficient) of the previous gradient steps. Using momentum in gradient descent, the weight update rule changes to

$$\begin{aligned} W &\mapsto W - \eta\nu_t \\ \nu_t &= \beta\nu_{t-1} + (1-\beta)\frac{\partial \mathcal{J}}{\partial W} \end{aligned} \tag{4.16}$$

where η is the learning rate as before, ν is called velocity and $\nu_0 = 0$, and $\beta \in [0, 1)$ is called momentum coefficient. The equation that describes the update on the velocity is an exponentially weighted average equation [8]. The first term in this equation takes a weighted average of the accumulated gradients in previous steps, while the second term considers the weighted gradient at the current step. Equation 4.16 will reduce to standard gradient descent for $\beta = 0$. By averaging over previous gradient steps with momentum, the gradient will become smaller in those directions that standard gradient descent might cause fluctuations. In the directions that gradient does not fluctuate, the average of gradient values is still large. Given a proper value of momentum, gradient descent with momentum can lead to a smoother and faster convergence toward a stable minimum. In

[6] We are talking about a high dimensional space, where each dimension corresponds with one parameter of the deep learning model. Not all parameters have the same effect on the loss function. For the same amount of variation in different parameters, the loss function can change quite differently. In other words, gradients of loss are different with respect to different parameters.

[7] Please refer to subsection 4.3.3.2

[8] This is called an exponentially weighted average equation because if we expand it, we see that at each time step, the current gradient is multiplied by $1 - \beta$, but the previous gradients are weighted by exponentially decaying values of β.

TABLE 4.2: Summary of the *MMGS* dataset for SGAR. Min frames and max frames show the minimum and maximum number of sequence frames in each class, respectively. Total frames shows the total number of frames from all the sequences in each class.

Class	min frames	max frames	Total frames
Normal	33	195	15726
Knee	41	216	19896
Padding	57	193	17180

general, $\beta = 0.9$ works well in many problems. We also use this value of momentum in each case where we utilize the SGD optimizer.

4.3.4 MMGS dataset

For this study, we use the multi-modal gait symmetry database (*MMGS*). This dataset was introduced in [5] by Khokhlova *et al.* *MMGS* contains three normal/pathological gait classes, simulated by healthy subjects. The gait classes include normal gait, limping gait that is simulated by wearing a 7-cm padding sole by each subject, and gait with knee injury-related problems. The latter category can represent an after-fracture recovering gait or a prosthesis-wearing gait and is simulated by asking subjects not to bend their right knee during walking. There are 27 subjects, each performing each of the gait categories between 5 to 7 times. The sequences in this dataset have different numbers of frames, ranging from 33 to 216 frames. Thus, different subjects have sequences of different lengths, and there exists class imbalance in this dataset. Table 4.2 summarizes the *MMGS* dataset, presenting the minimum and the maximum number of sequence frames, as well as the total number of frames in each class. As we observe from this table, the anomaly related to the knee injury has the maximum total number of frames among all the classes. We also see that normal class has the minimum number of total frames among the three classes. The numbers in this table confirm the class imbalance in this dataset.

Among the publicly available skeleton datasets for normal/pathological gaits, this dataset has the largest number of subjects. The large number of subjects in this dataset

and multiple records of the same walk results in high intra-class variations, which makes classification challenging for this dataset.

4.3.5 Experimental results

Table 4.3 shows classification scores of SGAR using the proposed features with different classifiers for the *MMGS* dataset. We use accuracy, precision, and recall (sensitivity) for classification evaluation. As we observe, compared with SVM, classification accuracy improves by almost 4% with the designed LSTM model. In Table 4.4, we present a comparison between the proposed feature vector and the work presented in [5] by Khokhlova *et al.*. They present low limb flexion angles as the feature vectors for SGAR. The low limb flexion angles in [5] is classified by an SVM with a polynomial kernel, a single bidirectional LSTM, and an ensemble of five bidirectional LSTMs as is reported in Table 4.4. Considering the results in Tables 4.3 and 4.4, we observe that with the proposed features, we acquired higher classification scores with Random Forest and SVM compared with the low limbs flexion with SVM and LSTM. Even without data augmentation, the proposed features can achieve better performance compared with both LSTM and ensemble of LSTMs in [5]. Our results also confirm the effectiveness of data augmentation in improving the performance of the LSTM model. With simple data augmentation, we can improve the average classification accuracy by more than 4% compared with an ensemble of LSTM networks in [5]. By comparing the minimum and maximum accuracy in the last two rows of Table 4.4, we see data augmentation can reduce accuracy variation of the model, resulting in a more robust model, as we expected.

In this study, we used the same subjects for testing as in the original study by Khokhlova *et al.* [5]. Out of 27 subjects, 8 of them were selected for testing and the rest for training and validation. From the remaining subjects, they used 5 subjects for validation and the rest for training. As a part of our experiments, we also investigated the effect of using different numbers of subjects for validation. Figure 4.10 shows average

TABLE 4.3: Average accuracy, precision, and recall (sensitivity) of gait anomaly recognition for the *MMGS* dataset using the proposed feature vector with five different classifiers. The results with LSTM is with data augmentation.

Model	**Accuracy**	**Precision**	**Recall/sensitivity**
KNN (N=5)	62.89	62.29	63.99
Naïve Bayes	70.20	69.63	69.73
Random Forest	79.33	79.21	79.37
SVM	81.87	81.22	81.80
LSTM	**85.63**	85.14	85.33

TABLE 4.4: Average accuracy, precision, and recall (sensitivity) of gait anomaly recognition for the *MMGS* dataset with the proposed feature vector and the low limbs flexion angles in [5]. The last two rows, labeled by $**$, show the results with our proposed feature and the designed bidirectional LSTM network. NA stands for **n**o **a**ugmentation.

Model	**Accuracy**	**Min accuracy**	**Max accuracy**
[5] SVM	77.6	-	-
[5] LSTM	77	71	94
[5] LSTM ensemble	82	75	91
LSTM** (NA)	83.67	71.54	89.89
LSTM**	**85.63**	76	91

classification accuracy as a function of number of subjects for validation over the range of $[3, 7]$ subjects. For the tested range of $[3, 7]$ subjects for validation, the average score doesn't change dramatically. However, the standard deviation of the average accuracy varies significantly. When we use 5 subjects for validation, the standard deviation of the scores over 30 runs of the network is the lowest, and for 6 subjects, the standard deviation reaches its highest value. We also observe that for 4 subjects, the average accuracy is the highest, while the standard deviation of the scores is slightly higher than the 5 subjects scenario. The reported results in Tables 4.3 and 4.4 shows the average classification scores with $N_{subjects} = 4$ for validation.

As we mentioned earlier, in the original study [5], the authors used two different sets of subjects for validation and training. While in gait anomaly recognition, it is essential to test on the subjects that the model was not trained on, using different subjects for training and validation is not required. Using data from more subjects for training can be beneficial, as the network will see a more diverse set of patterns per class during the training phase. Ultimately, with limited data, this can improve the generalization

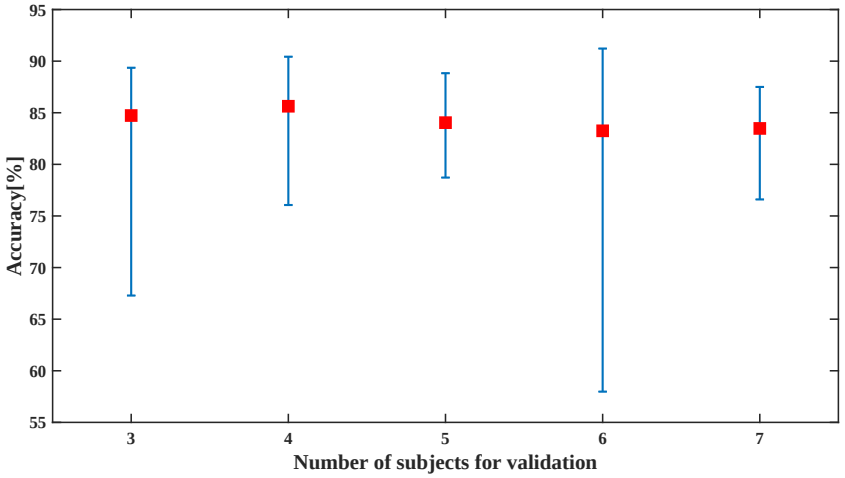


FIGURE 4.10: Effect of the number of subjects for validation on the average classification accuracy in the *MMGS* dataset.

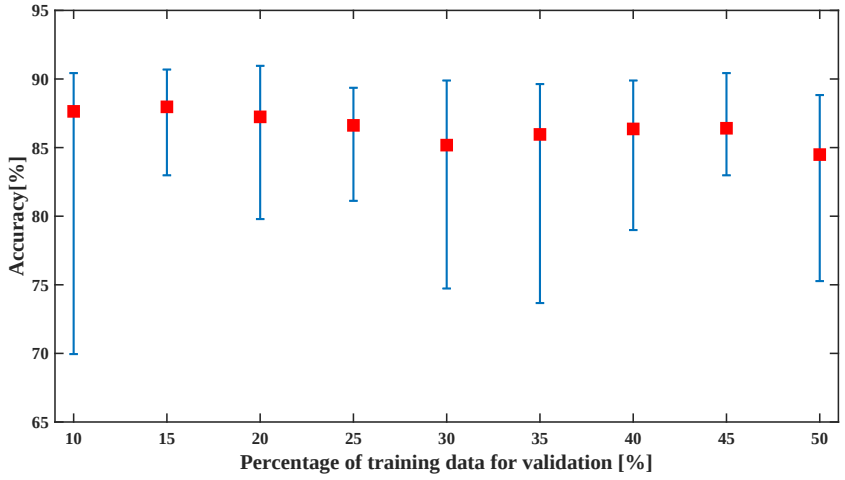


FIGURE 4.11: Percentage of training data for validation, based on the non-subject-based train-validation split for the *MMGS* dataset.

error of the network. Using a non-subject-based train-validation split criterion, we observe an increase in the classification scores. Figure 4.11 shows average classification accuracy using a non-subject-based train-validation split criterion. We computed the average classification accuracy over the range of $[10\%, 50\%]$ of the training data used for validation. With a non-subject-based criterion, we acquired an accuracy of 87.97% for three classes of normal/pathological gaits, using 15% of the training data for validation. This is an improvement of 2.34% over the best result that we achieved with the subject-based train-validation split and 7.97% improvement over the LSTM ensemble. Besides, we could improve the standard deviation of the network accuracy over multiple runs

(here 30 iterations) from 3.18% in the subject-based train-validation split to 1.88% for the non-subject-based train-validation split, that shows a more robust model in the latter case. As we mentioned earlier, when we utilize a subject-based train-validation split, we get the highest classification accuracy with 4 subjects for validation. Using 4 subjects for validation roughly corresponds with using 15% of the training data for validation in the non-subject-based train-validation split. We also observe that we acquire the highest classification accuracy with non-subject-based train-validation split when we use 15% of the training data for validation. These results suggest that for the *MMGS* dataset with the subject-based train-validation split, using 4 subjects for validation might be a better choice compared with the 5 subjects for validation that was originally suggested in [5].

4.4 SGAR with preprocessed skeleton joints

In the previous section, we presented a model for SGAR that was based on a set of handcrafted features. We designed an LSTM model for detecting and classifying the embedded patterns in different normal/pathological gait classes. We performed our experiments on the *MMGS* dataset, which includes sequences from 27 subjects with three classes of normal/pathological gaits. Relevant feature representation plays a pivotal role in any classification problem. Before the emergence of deep learning, feature descriptors in SGAR were mostly handcrafted. Such features are commonly learned through an unsupervised process. Handcrafted features can perform well in some applications, as they require domain knowledge. However, dismissing class information in the design process can also result in features that are domain-specific and cannot be generalized well. In contrast with the handcrafted features, deep learning models extract relevant features by considering class information. As a result, the extracted features are both pertinent and discriminatory.

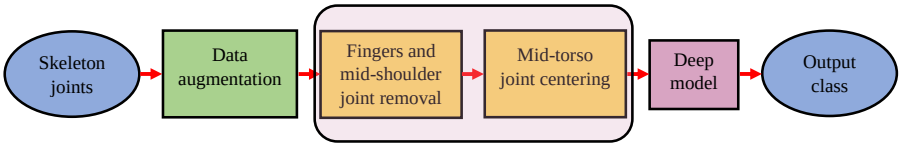


FIGURE 4.12: Pipeline for skeleton-based gait anomaly recognition, using minimally preprocessed 3D skeleton joints information. The knowledge of the trained model is then transferred to initialize similar networks for modeling of gait patterns in other datasets with different types of gait anomalies.

In general, in many classification problems, the input data exist in a high dimensional space. Traditional machine learning classifiers are not always proficient in learning meaningful features from high-dimensional inputs. With a feature representation, we create a set of relevant features in a low-dimensional space, such that a classifier can successfully distinguish between different classes. Given enough input samples, a well-designed deep learning model is capable of learning latent features embedded in the high-dimensional input data. Furthermore, studies have shown that given enough data, the learned features can be successfully transferred to relevant tasks [126].

Here, we present end-to-end feed-forward deep learning models for modeling and classification of spatiotemporal patterns of gait anomaly using minimally preprocessed skeleton data. Figure 4.12 describes the workflow of the presented gait anomaly recognition methodology. It starts with data augmentation. In the next step, a preprocessing procedure is performed that consists of removing certain joints and mid-torso joint centering. As can be seen from this figure, we remove fingers and mid-shoulder joints that are generally noisy. For mid-torso centering, we centralize the skeleton with respect to the mid-torso joint. This is accomplished by subtracting the coordinates of the mid-torso joint from the coordinates of each skeleton joint. Finally, we provide the resulting input to a deep learning model for gait anomaly classification. Besides, the trained models will be evaluated on other datasets through transfer learning. In the following subsections, we describe each of the steps in the presented pipeline.

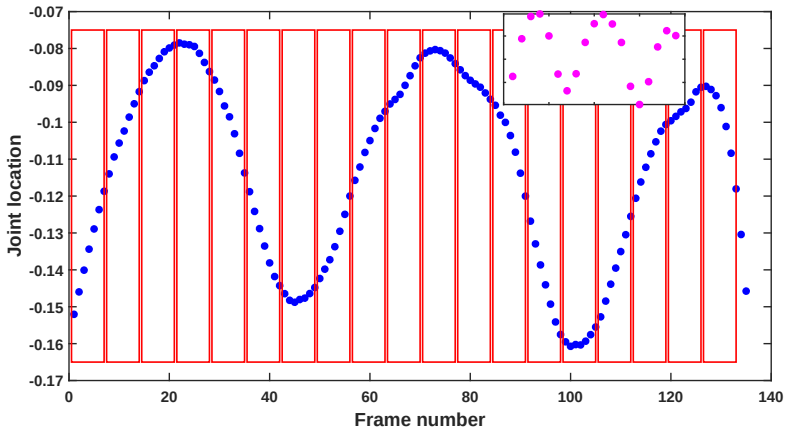


FIGURE 4.13: Data augmentation using the temporal moving mean for a sample joint coordinate sequence. Each value in the small plot on the top right of the figure shows the average of values within one of the red windows.

4.4.1 Data augmentation

Data augmentation is performed on the training data in three steps. In the first step, we use a temporal moving mean to compute the mean of each joint coordinate over a temporal window. This augmentation method is inspired by the local averaging in the spectral space in [127]. We compute the temporal moving mean over consecutive windows with no overlap. The resulting sequence can be presented by the following equation

$$s = [\mu_i]_{i=1}^{M} \qquad \mu_i = \frac{1}{k}\sum_{j=1}^{k} p_j \tag{4.17}$$

where μ_i is the mean over the ith temporal window, M is the total number of non-overlapping temporal windows in the original sequence, and k is the length of the temporal window that can be determined based on experimentation. p represents the joint coordinate in one of the x, y, or z directions, and $j = 1$ and $j = k$ mark the first and last element inside the ith moving window. By sorting the mean over each window in a timely order, we create a new sequence. Figure 4.13 illustrates data augmentation through temporal moving mean. Each red window in this figure shows one of the instances of the moving window. As can be seen, there is no overlap between consecutive instances of the

moving mean. We can see the newly created sequence in a small box at the top right corner of this figure. The next two steps of data augmentation are the same as what was described in subsection 4.3.2 for handcrafted features, except for the length of sequences. Here, all the sequences have the same length of 50 frames. This length has been selected following the same criterion as in [95]. To select the proper length, we use the approximate number of frames that can be collected in the reliable range of Kinect V2 when a subject walks in front of the sensor. Together, these three steps of data augmentation generate more data that can be used for training the deep learning models.

4.4.2 Preprocessing

In the preprocessing stage, following [3], we first remove all the finger joints along with the mid-shoulder spine joint from each skeleton, as they are noisy and do not offer useful information for detecting anomalies. The initial 25 joints of a Kinect skeleton is then reduced to 20 joints. Next, we translate the coordinates of each remaining joint by centralizing the skeleton with respect to the mid-torso joint. For a Kinect sequence with f frames, the translated 3-dimensional coordinates of the joints in frame i is represented by the following vectorized format

$$J_i = [x_k - x_{mt}, y_k - y_{mt}, z_k - z_{mt}]_{k=1}^{20} \in \Re^{3N} \tag{4.18}$$

where (x_k, y_k, z_k) and (x_{mt}, y_{mt}, z_{mt}) represent the joint coordinates of each of the remaining joints and the mid-torso joint coordinate in frame i, respectively. We will call the resulting preprocessed joints, ***selected normalized joints***. Finally, in order to have a model that can effectively learn the patterns in the input skeleton sequences, each feature (here each coordinate of the selected normalized skeleton joints) is rescaled into the $[-1, 1]$ range as was described in Equation 4.11.

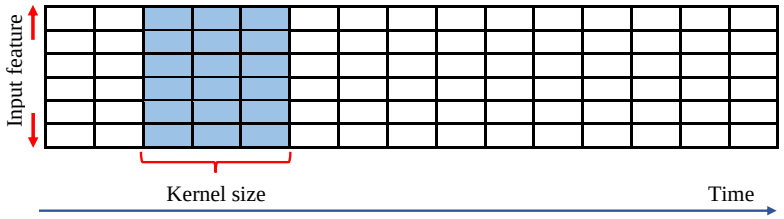


FIGURE 4.14: Illustration of how a one-dimensional convolutional kernel works, using an arbitrary input feature. In this figure, the kernel size is 3 and the input has 6 features. A 1D-convolutional kernel only moves in the direction of time (here from left to right).

4.4.3 Description of the models

We present end-to-end feed-forward deep learning models for modeling and classification of normal/pathological gait patterns. In particular, we compare the performance of three types of deep learning models: a fully convolutional network (FCN), a long short term memory network (LSTM), and a CNN-LSTM network. LSTM and one-dimensional CNN layers are principal parts of the presented models. We already described the mechanism behind a vanilla LSTM network. Before going into more detail about the architecture of the models, we will provide some background on one-dimensional CNN networks.

4.4.3.1 Some background on one-dimensional CNNs

During the last decade, convolutional neural networks have become the state-of-the-art in areas such as computer vision [128], natural language processing (NLP) [129, 130], image and video processing and analysis [131, 132], genomics [133], clinical data classification [134, 135], and finance [136]. Compared with their fully connected counterparts, CNN-based networks have fewer parameters. Furthermore, they are specialized in learning spatial features, with deeper layers in the network learning higher-level patterns in the data. 1D convolutional networks (1D CNN), which are used for time series analysis, have been popularized after the successful application of 1D CNN in the classification of ECG signals [137]. In the last few years, numerous 1D CNN architectures have been proposed for modeling univariate and multivariate time series problems [138–140]. In general, for

the time point t, the result of applying a 1D convolution with filter ω of length l on a univariate time sequence followed by a nonlinear function f such as ReLU, can be presented with the following [141]:

$$C_t = f(\omega \cdot X_{t-l/2:t+l/2} + b) \quad | \quad \forall t \in [1, T] \tag{4.19}$$

where b is the bias. Figure 4.14 shows how a 1D-convolutional kernel acts on an arbitrary input. As we can observe from this figure, compared to a 2D convolutional filter that moves in two directions, a 1D convolutional filter only moves along the time direction.

4.4.3.2 Transfer learning

In general, deep learning models work well on the dataset they have been trained on. However, they might not generalize well to new examples that raise conditions the model did not experience during the training. Transfer learning refers to the process of training a network on the source dataset and then using that network for a relevant task on another dataset. We adopt the description presented by Pan and Yang [142] for a mathematical definition of transfer learning that is based on the concepts of domain and task. A domain consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$. Given a domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task consists of a label space $\mathcal{Y}$ and an objective predictive function $f(.)$ ($\mathcal{T} = \{\mathcal{Y}, f(.)\}$), where $f(.)$ is learned from the training data consisting of pairs $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.

Definition of transfer learning: Given a source domain $\mathcal{D}_S$ and its corresponding learning task of $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and its learning task of $\mathcal{T}_T$, the goal of transfer learning is to improve the learning of the target predictive function $f_T(.)$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.

The application of the pre-trained network on the target dataset can be in the form of feature extraction or fine-tuning. In feature extraction, we change the final layer of the

network that provides the probability of each class. All or some parts of a model that was trained on the source dataset are used for feature extraction, meaning the weights of them are retained. Next, one or more layers are added for class prediction in the target domain. During the training phase, only the newly added layers are trained. This approach is more suitable when the target and source domains are similar and the dataset in the target domain is small. In fine-tuning, the final layer of the pre-trained network is changed. The network is initialized with the weights of the pre-trained model, and the weights are fine-tuned with the target dataset, generally with a smaller learning rate. In fine-tuning, we can freeze some earlier layers of the network that are responsible for the high-level features and fine-tune only the deeper layers. Transfer learning can be in particular useful for smaller datasets, where training on a larger related collection of observations can boost the performance of the model on a smaller dataset.

While transfer learning has been successfully employed in numerous natural language processing [143,144], computer vision [145,146], and image and video processing [147–150] problems, the application of transfer learning has remained mostly limited for time-series data [151]. As is discussed in [151], one of the main challenges in employing transfer learning for time series datasets is finding a properly-related set of datasets as the source and target. With the availability of smaller datasets in time series-based classification problems, finding the proper source and target datasets for transfer learning can further reduce the generalization error of deep models for such applications. Considering SGAR as a time series-based problem, the majority of studies in SGAR focus on one dataset, and to the best of our knowledge, transfer learning has never been employed in any SGAR study.

4.4.3.3 Architecture of the deep learning models

In this subsection, we describe the architecture of the different deep learning models that we designed for SGAR. We explain the structure of the CNN-LSTM model that we used

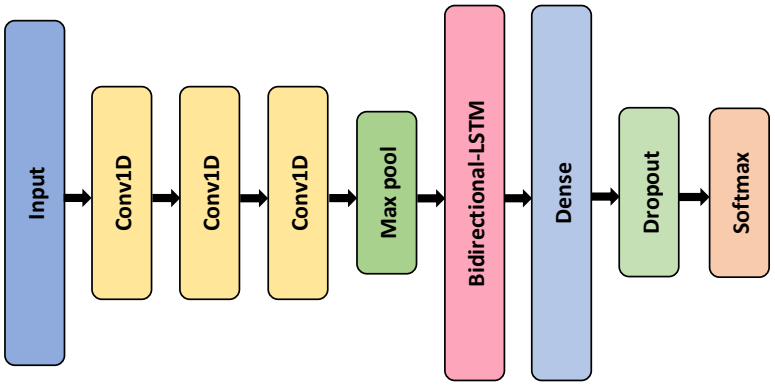


FIGURE 4.15: Structure of the CNN-LSTM for modelling and classifying selected normalized joints. Each conv1D block consists of one-dimensional convolutional layers followed by a ReLU activation function.

for modeling and classification of the selected normalized joints in more detail. Other networks will be described more briefly as the structures are straightforward, and there are similarities in terms of the number of inputs, number of outputs, weight initialization, optimizer, loss function, and some other settings.

CNN-LSTM model for selected normalized joints: Figure 4.15 shows the structure of the designed CNN-LSTM model. The designed model is a sequential CNN-LSTM, which is trained by mini-batches of 32 samples. Each mini-batch consists of sequences of selected normalized joints. Each sequence has a dimension of $R^{50\times60}$, with a sequence length of 50 and 60 features per time step corresponding with 20 normalized joint location coordinates in three dimensions (refer to Equation 4.18).

Each convolutional layer has 32 neurons with a kernel size of 5, followed by a rectified linear unit (ReLU) as the nonlinear transformation. The output of each convolutional layer is a matrix of $R^{50\times32}$, where each column of this matrix contains the weights of one filter. The convolutional layers use *same* padding which means the size of the output feature map of each layer is the same as the size of its input feature map. This is also equivalent to $stride = 1$, where stride is the amount of movement of the kernel on the input feature map between any two successive applications of the kernel. Figure 4.16 provides an illustration of stride and zero padding for a sample input with a sequence of

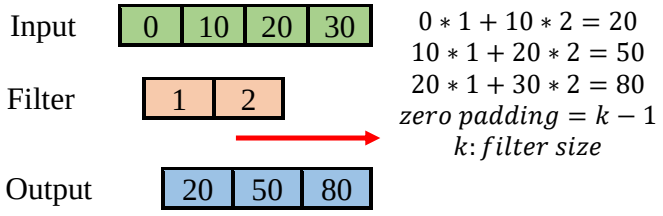


FIGURE 4.16: Illustration of zero-padding and stride for a kernel of $size = 2$ and a sample input sequence of $length = 3$, where the zero at the beginning of the input sequence is used for zero-padding to create an output of the same length as the input sequence. The filter moves from left to right (in the direction of the red arrow), acting on the input elements to create the output elements.

length 3 and a 1D convolutional kernel of $size = 2$. The zero at the beginning of the input sequence is for zero-padding that is required for generating an output of the same length as the input. As we observe in this figure, the filter moves from left to right and acts on the input to create the output.

After the 3rd convolutional layer, we use a one-dimensional max pooling layer with a pool size of 2 that acts along the time direction and creates outputs of size $R^{25\times32}$. The max pooling layer is followed by a single bidirectional LSTM layer, with 100 neurons and a recurrent dropout of 50% probability that creates a feature map of $R^{1\times100}$. Through a random dropout of the neurons in the LSTM layer, the likelihood of different neurons affecting one another reduces, and the resulting network is less sensitive to smaller variations in the data. After the bidirectional layer, there is a fully connected layer of 100 neurons with ReLU activation, followed by a dropout layer of 30% probability. Finally, there is a softmax layer for classification. To reduce the generalization error, apart from the dropout, we also use $\mathcal{L}1$-norm for the kernel regularization. The Kullback-Leibler divergence is adopted as the loss function

$$D_{KL}(p \parallel q) = \sum_{i=1}^{N} p(x_i) \log \frac{p(x_i)}{q(x_i)} \tag{4.20}$$

where x_is are the observations, and p and q are the true and predicted distributions,

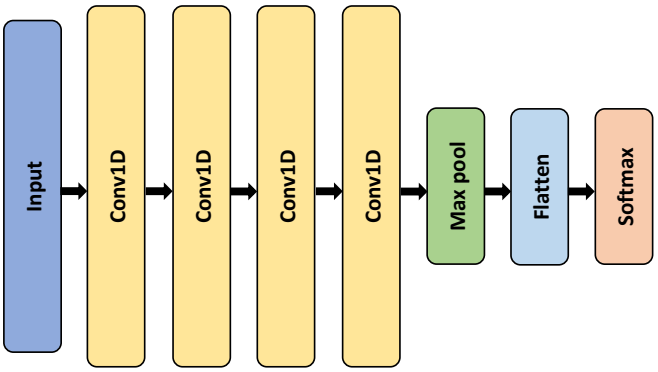


FIGURE 4.17: Structure of the FCN for modelling and classifying selected normalized joints.

respectively. The weights in all the layers are initialized using Glorot's uniform initialization [152], where weights are sampled uniformly from the $[-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}]$ interval. Here, n_{in} and n_{out} are the number of input and output units to the weight tensor. The network is trained for 400 epochs, uses early stopping with $patience = 220$, and SWA is initialized at $epoch = 211$.

TABLE 4.5: Hyperparameters of the designed FCN for SGAR using the *Walking gait* dataset and selected normalized joints. The number of layers refers to the number of convolutional layers. The network also includes a max pooling layer of pool size 2, and a flatten layer to concatenate the two-dimensional arrays into the one-dimensional arrays required by the softmax for classification.

Hyperparameter	Value	Hyperparameter	Value
Number of layers	4	SWA learning rate	0.0003
Number of units	650	Epochs	400
Batch size	32	Optimizer	SGD
Learning rate	0.0007	Loss function	KL-divergence

FCN model for selected normalized joints: Figure 4.17 shows the architecture of the designed FCN model for modeling and classifying normal/pathological gait categories using selected normalized joints as the input features. Like before, each feature is scaled in the $[-1, 1]$ range before feeding to the network. The network has four one-dimensional convolutional layers followed by a max pooling layer with a pool-size of 2. Each convolutional block uses kernels of size 3, ReLU activation, and *same* padding [9]. Convolutional layers also use $\mathcal{L}1$ kernel regularization. The number of epochs, the epoch

[9]All the 1D convolutional layers in this thesis use *same* padding and are followed by ReLU as the activation function, unless otherwise stated

at which SWA is initialized and patience are all the same as the CNN-LSTM network above. We summarized some of the key hyperparameters in the designed FCN model in Table 4.5.

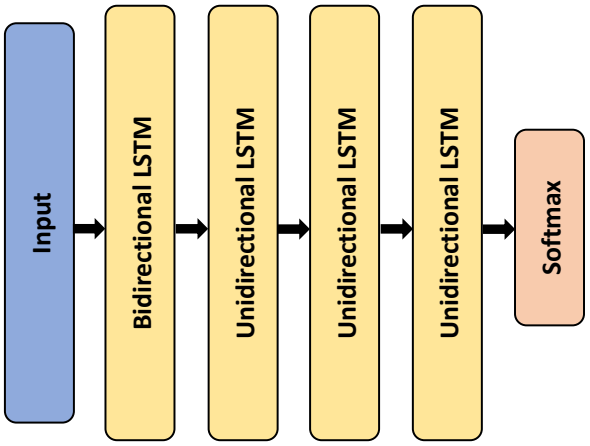


FIGURE 4.18: Structure of the LSTM for modelling and classifying selected normalized joints. The bidirectional layer merges its outputs by taking their average at each time step.

LSTM model for selected normalized joints: Figure 4.18 shows the structure of the designed LSTM model for modeling and classifying normal/pathological gait categories using selected normalized joints as the input. The network has four LSTM layers, with the first layer being bidirectional and the next three LSTM layers are unidirectional. Each LSTM layer uses $\mathcal{L}1$ kernel regularization. The patience for early stopping and SWA initialization both occur at the same epochs as in CNN-LSTM and FCN networks. Table 4.6 shows the key hyperparameters in the designed LSTM model.

TABLE 4.6: LSTM network hyperparameters for SGAR using the *Walking gait* dataset with selected normalized joints as the input.

Hyperparameter	Value	Hyperparameter	Value
Number of layers	4	SWA learning rate	0.001
Number of units	64	Epochs	400
Batch size	32	Optimizer	SGD
Learning rate	0.005	Loss function	KL-divergence

CNN-LSTM model for leg angles: Figure 4.19 shows the structure of the designed CNN-LSTM network for SGAR using leg angles proposed in [2] as the input features. Each input feature is scaled in the $[0, 1]$ range according to Equation 4.11 before

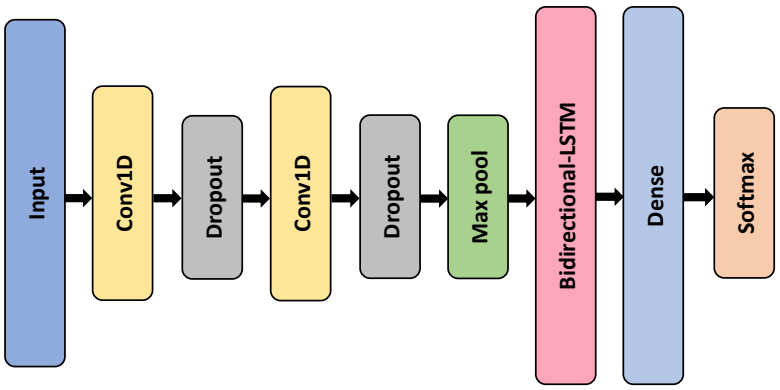


FIGURE 4.19: Structure of the CNN-LSTM for modelling and classifying leg angles features [2].

TABLE 4.7: Hyperparameters of the CNN-LSTM for SGAR using leg angles [2] as the feature for the *Walking gait* dataset.

Hyperparameter	Value	Hyperparameter	Value
Epochs	400	SWA learning rate	0.001
Batch size	32	Optimizer	SGD
Learning rate	0.005	Loss function	Cross-entropy

being passed to the network. Each conv1D has 32 one-dimensional convolutional filters of length 3. There is a dropout layer of probability 40% after each convolutional block that acts as a regularizer to reduce overfitting of the network. A max pooling of *pool-size* $= 2$ is located after the second dropout layer to reduce the dimension of the feature map in the time direction. The max pooling is followed by a bidirectional LSTM layer with 100 units and a recurrent dropout of 50% to capture the spatiotemporal patterns. A dense layer of 100 neurons comes after the bidirectional LSTM that passes its output to the softmax layer for classification of normal/pathological gaits. We summarized the main hyperparameters of this network in Table 4.7.

FCN model for leg angles: Figure 4.20 shows the structure of the designed FCN network for SGAR using the leg angles [2] as the input. Before feeding each example to the network, each feature is scaled in the $[-1, 1]$ range. The network has three one-dimensional convolutional layers, each followed by a dropout of 40% probability. Each convolutional layer uses kernels of size 3 and $\mathcal{L}2$ kernel regularization. There is a global max pooling layer after the 3rd convolutional layer and a dropout. Finally, the global max pooling layer is followed by a dropout layer of 35% probability and a softmax layer for

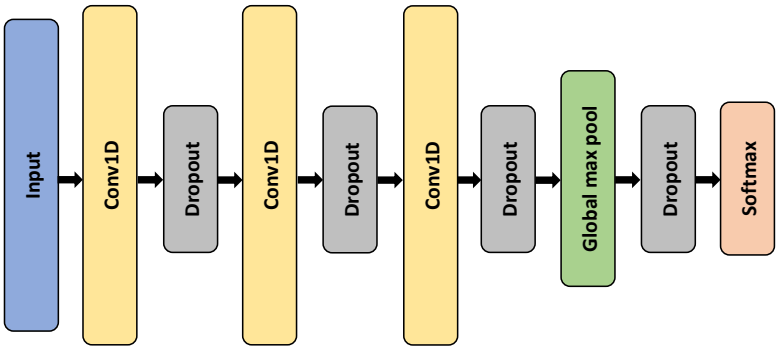


FIGURE 4.20: Structure of the FCN for modelling and classifying leg angles features [2].

TABLE 4.8: Key hyperparameters of the FCN network for SGAR using leg angles [2] as the feature for the *Walking gait* dataset. The number of layers only refers to the number of 1D convolutional layers in this network.

Hyperparameter	Value	Hyperparameter	Value
Number of layers	3	SWA learning rate	0.0001
Number of units	64	Epochs	400
Batch size	16	Optimizer	Adam
Learning rate	0.0007	Loss function	Cross-entropy

classification. The number of epochs, patience, and the epoch at which SWA is initialized are all the same as the CNN-LSTM model for 7 Angles that was described above. The key hyperparameters of the FCN model are presented in Table 4.8. Among the deep learning models that were designed for the *Walking gait* dataset, this is the only network that uses an optimizer other than SGD. In the following paragraphs, we will briefly describe how Adam works. Adam takes advantage of two main ideas in its design. Part of the concepts behind Adam is based on RMSprop, another gradient-based optimizer. In the following paragraphs, first, we will explain how RMSprop works. Next, we will describe the idea behind Adam.

RMSprop: Root mean square propagation (RMSprop) [153] is a gradient-based optimization method that was proposed by Geoffrey Hinton for mini-batch learning. It uses different learning rates for different parameters of the network, and each of these learning rates adapts individually over time. At each iteration, the weights are updated

according to the following equation

$$\begin{aligned} W &\mapsto W - \frac{\eta}{\sqrt{\nu_t} + \epsilon} \frac{\partial \mathcal{J}}{\partial W} \\ \nu_t &= \rho \nu_{t-1} + (1 - \rho)(\frac{\partial \mathcal{J}}{\partial W})^2 \end{aligned} \tag{4.21}$$

where ρ is the moving average parameter, η is the learning rate, $\frac{\partial \mathcal{J}}{\partial W}$ is the partial derivative of cost function with respect to weight, ν_t is the exponential average of the squares of gradient, and ϵ is a very small constant for numerical stability. By normalizing the learning rate by the root of squared gradient, RMSprop reduces the learning rate in the directions with large fluctuations (large gradient value) and increases learning rate in the directions with small gradients. Similar to momentum in SGD, RMSprop can reduce the fluctuations in the directions with large-amplitude gradient oscillations and accelerate the learning process.

Adam: Along with RMSprop, adaptive moment estimation (Adam) [154] is another adaptive learning-based optimization technique that has been applied successfully in a wide range of different deep learning architectures. Adam combines RMSprop and momentum together to update the parameters of the model at each iteration. Just like momentum, Adam keeps an exponentially decaying average of the past gradients, and like RMSprop it reserves an exponentially decaying average of the previous squared gradients. So, for each parameter of the model, Adam computes the following two parameters

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial \mathcal{J}}{\partial W} \\ \nu_t &= \beta_2 \nu_{t-1} + (1 - \beta_2)(\frac{\partial \mathcal{J}}{\partial W})^2 \end{aligned} \tag{4.22}$$

where m_t and ν_t are estimates of the first and second moments (the mean and variance, respectively). As m and ν are initialized as zero, the authors of Adam recognized that m and ν tend to be biased towards zero. Therefore, they performed a bias-correction such

TABLE 4.9: Hyperparameters of the LSTM network for the *Walking gait* dataset using leg angles [2] as the input. The first and second LSTM layers are bidirectional and unidirectional, respectively.

Hyperparameter	Value	Hyperparameter	Value
Number of layers	2	SWA learning rate	0.0005
Number of units	256	Epochs	500
Batch size	64	Optimizer	SGD
Learning rate	0.001	Loss function	Cross-entropy

that

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{\nu}_t &= \frac{\nu_t}{1-\beta_2^t} \end{aligned} \tag{4.23}$$

Finally, the weights are updated according to the following

$$W \mapsto W - \frac{\eta}{\sqrt{\hat{\nu}_t} + \epsilon}\hat{m}_t \tag{4.24}$$

Like before, η is the learning rate, and ϵ is a small number for numerical stability. In the original paper, the authors proposed $\beta_1 = 0.9$ and $\beta_2 = 0.999$. These values of β_1 and β_2 work quite well in most of the problems.

LSTM model for leg angles: Table 4.9 provides a description of the LSTM network for modeling and classifying normal/pathological gait categories using leg angles [2] as the input. Each of the input features is scaled in the $[0, 1]$ range following Equation 4.11. The network has two LSTM layers, where the first and second layers are bidirectional and unidirectional, respectively. The second LSTM layer is followed by a softmax layer for classification. Both layers use $\mathcal{L}2$ kernel regularization. The network utilizes early stopping with *patience* $= 270$ and SWA initialization happens at *epoch* $= 266$.

TABLE 4.10: Hyperparameters of the CNN-LSTM for the *Walking gait* dataset using distance between joints [3] as the input.

Hyperparameter	Value	Hyperparameter	Value
Epochs	80	SWA learning rate	0.0007
Batch size	32	Optimizer	SGD
Learning rate	0.003	Loss function	KL-divergence

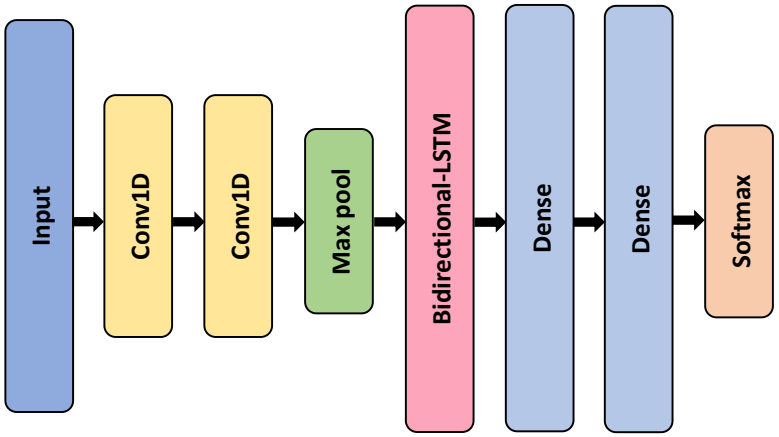


FIGURE 4.21: Structure of the CNN-LSTM for gait anomaly classification using the distance between skeleton joints [3].

TABLE 4.11: The hyperparameters of the FCN network with the distance between joints [3] as the input for the *Walking gait* dataset. The number of layers only points to the number of 1D convolutional layers. There are 256 units (neurons) in the first two convolutional layers, and the last convolutional layer has 128 neurons.

Hyperparameter	Value	Hyperparameter	Value
Number of layers	3	SWA learning rate	0.0003
Number of units	256/128	Epochs	80
Batch size	16	Optimizer	SGD
Learning rate	0.003	Loss function	Cross-entropy

CNN-LSTM model for distance between joints: Figure 4.21 shows the structure of the designed CNN-LSTM network for SGAR using distance between joints feature [3] as the input. Each input feature is scaled in the $[-1, 1]$ range according to Equation 4.11 before being passed to the network. There are two 1D convolutional layers, each with 100 one-dimensional convolutional filters. The first and second convolutional layers have filters of length 3 and 1, respectively. A max pooling of *pool-size* $= 2$ is located after the second convolutional block, followed by a bidirectional LSTM layer with 100 kernels and a recurrent dropout of 50%. There are two dense layers of 100 neurons that come after the bidirectional LSTM. Both dense layers use ReLU activation function. Finally, there is a softmax layer for the classification of normal/pathological gaits. The layers use $\mathcal{L}1$ kernel regularization. The network is trained for 80 epochs, with early stopping patience of 50 and SWA initialization at *epoch* $= 46$. We summarized the main hyperparameters of this network in Table 4.10.

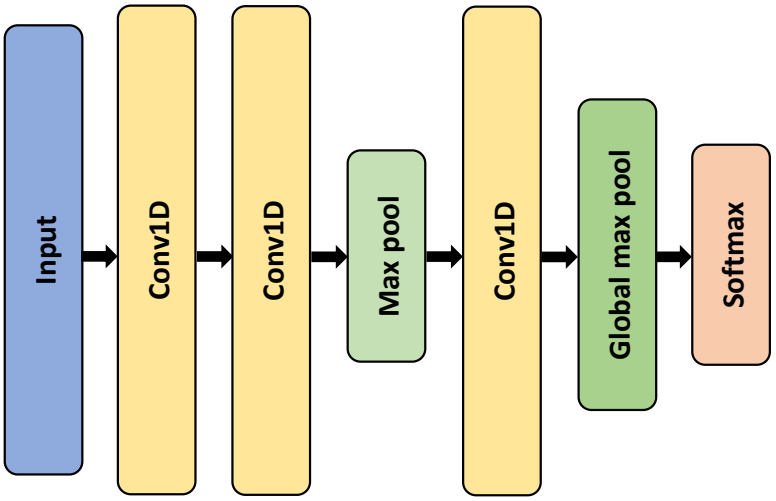


FIGURE 4.22: Structure of the FCN for modeling and classifying normal/pathological gait patterns, using the distance between skeleton joints [3] as the input.

FCN model for distance between joints: Figure 4.22 shows the structure of the designed FCN network using the distance between joints [3] as the input feature. Before feeding any input to the network, each feature is scaled in the $[-1, 1]$ range. The network has three one-dimensional convolutional layers, all using $\mathcal{L}1$ kernel regularization. The first and third convolutional layers use kernels of size 3, but the second conv1D uses kernels of size 1. There is a global max pooling layer after the 3rd convolutional layer followed by a softmax layer for classification. The number of epochs, patience, and the epoch at which SWA is initialized are the same as the CNN-LSTM model that was described above. The main hyperparameters of the FCN model are presented in Table 4.11.

TABLE 4.12: Description of the LSTM network using the distance between joints [3] as the feature vector for the *Walking gait* dataset. There are two LSTM layers and one dense layer in this network. The first and second LSTM layers are bidirectional and unidirectional, respectively. The number of units refers to the number of kernels in the LSTM layers only.

Hyperparameter	Value	Hyperparameter	Value
Number of layers	3	SWA learning rate	0.0003
Number of units	512	Epochs	350
Batch size	16	Optimizer	SGD
Learning rate	0.0009	Loss function	KL-divergence

LSTM model for distance between joints: Table 4.12 provides a description of the LSTM network for SGAR using distance between joints [3] as the input features. Each of the input features is scaled in the $[-1, 1]$ range following Equation 4.11. The

network has two LSTM layers, where the first and second layers are bidirectional and unidirectional, respectively. The second LSTM layer is followed by a dense layer with 100 kernels. Finally, a softmax layer is used for classification. The LSTM and dense layers all use $\mathcal{L}1$ kernel regularization. The network utilizes early stopping with *patience* = 175 and SWA initialization happens at *epoch* = 170.

Next, we will describe the *Walking gait* and *Pathological gait* datasets and then present experimental results.

TABLE 4.13: Description of each class in the *Walking gait* dataset

Class	Class description
Gait 1	Normal
Gait 2	Padding a 5 cm thick sole under left foot
Gait 3	Padding a 10 cm thick sole under left foot
Gait 4	Padding a 15 cm thick sole under left foot
Gait 5	Weight of 4 kg on the left ankle
Gait 6	Padding a 5 cm thick sole under right foot
Gait 7	Padding a 10 cm thick sole under right foot
Gait 8	Padding a 15 cm thick sole under right foot
Gait 9	Weight of 4 kg on the right ankle

4.4.4 Walking gait dataset

Nguyen et al. [155] collected a dataset containing nine classes of normal/pathological gaits. They asked subjects to wear padded soles of 5, 10, and 15 cm in thickness. Each subject wore each of the padded sole once on the right foot and once on the left foot. To collect more classes of anomalies, subjects also wore a 4-kg weight ankle once on each ankle. In total, there are nine subjects, each performing eight abnormal and one normal walk, where every sequence contains 1200 frames. This means there are 10800 frames per class. Apart from the work of Jun *et al.* [95], all the SGAR studies on the *Walking gait* dataset have used a two-class framework, which considers all the anomalous gait types as one class. However, as we mentioned earlier in section 4.1 in the work of Jun *et al.*, there is cross-over between the training and test subjects. Table 4.13 shows a description

TABLE 4.14: Description of five pathological gait categories in the *Pathological gait* dataset [4]

Pathological gait	Characteristics	Causes
Antalgic gait	Attempt to keep weight off the injured leg to avoid pain, shortening the stance phase of the injured leg	Pain in the foot, ankle, knee, or hip
Stiff-legged gait	Stiffness of the problematic leg while walking, making an outward semicircle while swinging the problematic leg	Joint-related pathologies, such as rheumatoid arthritis
Lurching gait	lurching the trunk backward at the heel strike of the problematic leg to compensate for weakness of hip extension	Weakness or paralysis of the gluteus maximus muscle
Steppage gait	Dorsiflexion problem in the problematic leg, lifting the problematic leg higher than normal to keep the toes from scraping the ground	Weakness or paralysis of the anterior tibialis muscle
Trendelenburg gait	Moving the problematic hip up and the opposite hip down during the stance phase to balance the hip level, lurching the trunk toward the problematic side	Weakness or paralysis of gluteus medius and gluteus minimus

of each of the gait classes in the *Walking gait* dataset based on the way each category is simulated by the healthy subjects.

4.4.5 Pathological gait dataset

Pathological gait dataset was collected by Jun *et al.* [4] using a system of six calibrated Kinect sensors. The sensors have been calibrated to acquire the same XYZ coordinate to collect consistent data from different directions. Ten healthy subjects simulated five categories of pathological gaits: antalgic, stiff-legged, lurching, steppage, and Trendelenburg gaits. Table 4.14 describes each class of the pathological gaits, along with examples of each type of abnormality.

Each subject repeated each class of gait 20 times. Considering the normal gait, this dataset contains six classes of normal/pathological gaits. Therefore, 120 sequences were collected from each subject. Among the publicly available datasets for SGAR, this dataset has the largest number of frames per class and the largest total number of frames. Table 4.15 presents a summary of the number of frames in each class of the *Pathological gait* dataset. As we see, the lurch gait is the largest class in this dataset that has more than twice the number of frames as in the normal gait, the class with the smallest number of

frames in this dataset. We also see that sequences do not have the same length. Therefore, like *MMGS*, this dataset also has a class imbalance.

TABLE 4.15: Summary of the *Pathological gait* dataset for SGAR. Min frames and max frames shows the minimum and maximum number of sequence frames in each class, respectively. Total frames shows the total number of frames from all the sequences in each class.

Class	min frames	max frames	Total frames
Normal	42	100	80972
Antalgic gait	51	218	142208
steppage	56	212	131130
lurch	58	314	167956
Stiff-legged	58	231	133281
Trendelenburg	58	230	156809

4.4.6 Experimental results

For this study, one goal is to design a framework that can be extended to other types of gait anomalies. To investigate the transferability of the proposed framework, we use transfer learning. In the first step, we perform a multi-class SGAR on the source dataset. Next, we use the network that was pre-trained on the source dataset to predict classes in the target datasets via transfer learning techniques. While *Pathological gait* is the largest publicly available dataset for SGAR, *Walking gait* has the largest number of normal/pathological gait classes, which means it covers a higher range of minor gait abnormalities. Besides, in the *Walking gait* dataset, each subject contributes only one sequence per class, which, of course, yields less intra-class variation. Based on these properties, we select *Walking gait* as the source dataset for transfer learning. Therefore, evaluation on the *MMGS* and *Pathological gait* datasets is performed through transfer learning.

4.4.6.1 Performance evaluation on Walking gait

To evaluate the efficacy of the proposed model for multi-class minor gait recognition, we compared the performance of our method with the features presented in [2] and [3]. In [2], authors use seven angles in the lower body for abnormal gait recognition. In [3],

TABLE 4.16: Average recognition accuracy and F-score for the *Walking gait* dataset using selected normalized joints (proposed), leg angles [2], and distance between joints [3]. For comparison, we look at the performance of three deep learning and three non-deep learning classifiers (LR: logistic regression and RF: random forest).

Input	Score	LR	RF	SVM	LSTM	FCN	CNN-LSTM
Leg angles [2]	Accuracy	53.76	67.16	63.27	65.59	65.56	63.83
	F-score	53.99	68.90	65.65	66.82	66.75	64.65
Distance between joints [3]	Accuracy	42.45	25.67	50.93	57.31	58.34	52.47
	F-score	47.68	39.54	53.34	60.83	62.42	55.30
Selected normalized joints	Accuracy	**70.04**	**68.47**	**74.42**	**78.61**	**88.52**	**90.57**
	F-score	**71.33**	**69.66**	**74.59**	**79.29**	**88.81**	**90.90**

Meng *et al.*. utilize the distance between each pair of skeleton joints, excluding the fingers and mid-shoulder joints, for gait anomaly recognition. Selecting these two features for comparison is based on the following assumptions. A lot of clinical gait assessment methodologies are based on the features from the lower body, which suggests these types of features are discriminative attributes for gait anomaly recognition. On the other hand, distance between skeleton joints considers a relation (here, Euclidean distance) between each pair of skeleton joints. Different body parts interact with one another during motion. Thereby, distance between skeleton joints that describes a relationship between different body parts can also be a proper candidate for comparison. For a fair comparison, we designed three deep learning models for the features in each of these studies. Besides, all the networks have around the same number of layers, where each structure has at most five layers (here we do not consider dropout, pooling, and flattening layers). We also compare the performance of each feature with three other classifiers: support vector machine (SVM), logistic regression (LR), and random forest (RF). For these three classifiers, we concatenate feature vectors over the same time window that we use as the sequence length of the input in the deep learning models. The performance evaluation is conducted on both datasets.

For each classifier, we train the model 15 times and report the average test accuracy and F-score. We use the sequences of 6 subjects for training and the remaining 3 subjects for test. We use 20% of the training data for validation for all of the cases. Table 4.16

shows the performance of each feature with six different classifiers. To avoid overfitting, we use early stopping in all of the deep learning models. The results show that our proposed feature (selected normalized joints) outperforms the other input features with all the classifiers. Among the features, distance between the joints [3] acquires minimal improvement with the deep learning models. One reason for the poor performance of this feature can be its high dimension (180 features per frame) that might require a more complex model and further tuning to attain higher classification scores. For the proposed features, CNN-LSTM outperforms the other models. But, considering all the competing features, on average, the FCN achieves the best classification scores among all the models. The results also show that among the deep learning models, the LSTM networks acquire the lowest scores.

Handcrafted features offer an interpretable set of descriptors for a classification task. However, their performance is more likely to be affected by the quality of new datasets. Previous studies have suggested as the condition under which data is collected changes, methods that employ handcrafted features might require further preprocessing [156]. Therefore, handcrafted features that perform well in one problem might not achieve reasonable performance in another problem, or they might need proper handling. Our results suggest that a deep network that learns the relevant features for maximal class distinction outperforms those models that employ handcrafted features for abnormal gait classification. Among the competing features, deep learning models that take *selected normalized joints* as the input, yield the maximum improvement relative to the performance of non-deep learning models.

In the next section, we will take a look at the CNN-LSTM model that was designed for selected normalized joints and achieved the highest prediction accuracy among other models. We will perform an ablation study to investigate the effect of different parts of the network on the performance of the model. This can be helpful in designing simpler

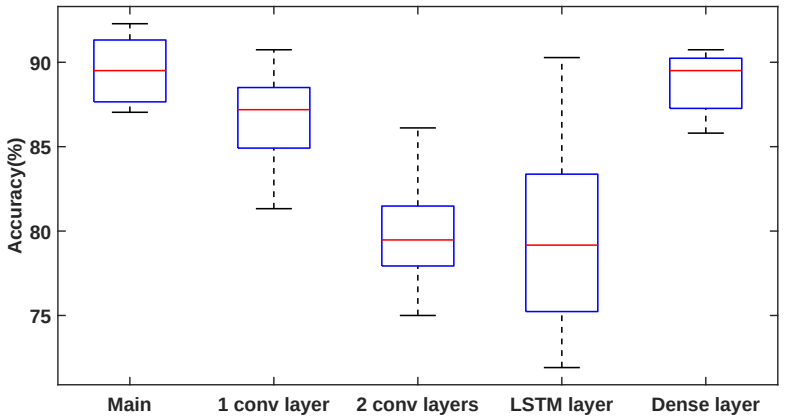


FIGURE 4.23: Ablation study on the CNN-LSTM layers: boxplots of the average classification accuracy for the *Walking gait* dataset. Each boxplot represents the average classification accuracy acquired by removing part of the CNN-LSTM network.

models, while still achieving high prediction scores. This study is important because, in general, models that are simpler are less likely to overfit and can generalize better to the unseen data.

4.4.6.2 Ablation study on CNN-LSTM layers

We performed an ablation study to investigate the effect of different layers in the classification performance of the designed CNN-LSTM network. Figure 4.23 shows the boxplot of the classification accuracy for the original CNN-LSTM and structures that are the result of excluding different layers. By comparing the boxplots in Figure 4.23, we see that removing the dense layer has the least effect on the performance of the network, reducing the average accuracy by about 2%. Eliminating one and two convolutional layers degrade accuracy by about 3.68% and more than 10%, respectively. Removing the LSTM layer has the most significant effect on the performance of the network with more than 11% reduction on the average classification accuracy. Besides, removing LSTM also results in a large variation on the average classification accuracy over multiple runs of the network, which suggests a larger generalization error. This observation further stresses the significance of the LSTM layer in this network.

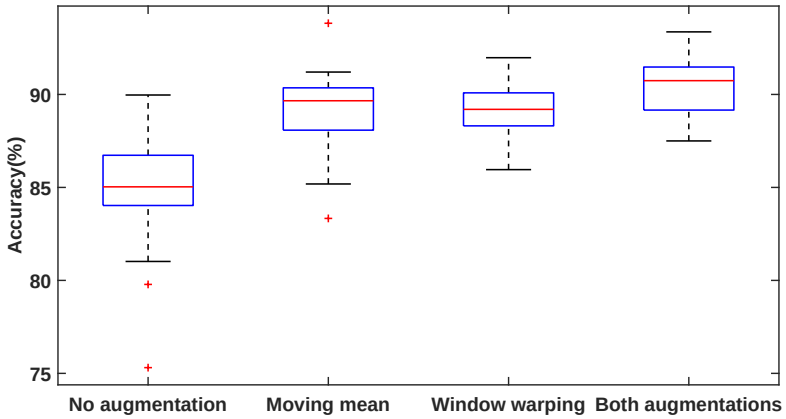


FIGURE 4.24: Boxplots of average classification accuracy for the *Walking gait* dataset, with and without augmentation. "Both augmentations" refers to the scenario where both "temporal moving mean" (in the figure "moving mean") and "window warping" are used for augmentation.

The results of this study can be helpful in transfer learning. For example, since the dense layer does not have a significant contribution to the performance of the model, if overfitting occurs during transfer learning, one way to improve the performance of the model is to remove the dense layer. In the next section, we will investigate the effect of data augmentation on the performance of the model. In particular, we are interested to see whether data augmentation could improve the classification accuracy of a deep model. for this study, we will once again use the CNN-LSTM model that was designed for the selected normalized joints.

4.4.6.3 Effect of data augmentation

The boxplots in Figure 4.24 illustrate the effect of data augmentation on classification accuracy of *Walking gate* dataset. As we can see in this figure, both *temporal moving mean* and *window warping* improve the average classification accuracy. The results also show that while *temporal moving mean* augmentation can result in some outlier scores, it improves the average accuracy the most. With *window warping*, we get less improvement; but, the classifier demonstrates the least variations over multiple runs of the model. Finally, by combining both augmentation methods, the model can acquire higher classification scores compared with the "no augmentation" case, without generating outlier

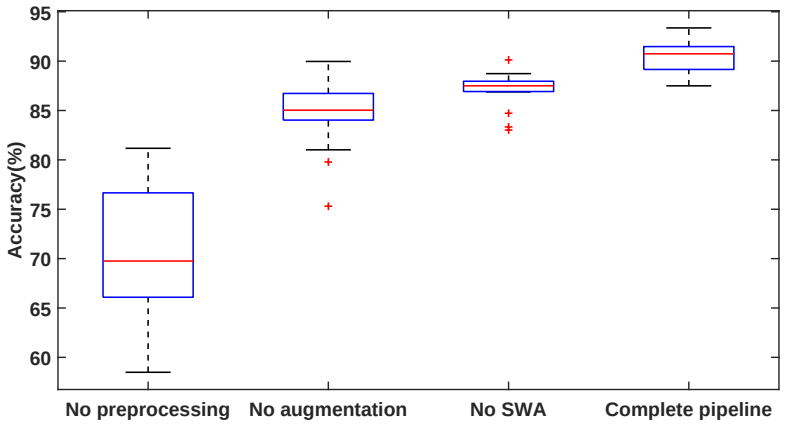


FIGURE 4.25: Boxplots of average classification accuracy for ablation study on the pipeline with the *Walking gait* dataset. Each boxplot represents average classification accuracy acquired by removing one step in the pipeline.

scores over several iterations of training. These results suggest the benefits of data augmentation for the SGAR problem in terms of improvement in prediction accuracy and variance of scores. The lower variance also indicates the resulting model is more robust, learning features with higher generalization potential. The results of this experiment also suggest that temporal moving mean and window warping are both proper augmentation methods for the SGAR problem.

So far, we performed an ablation study on a deep learning model and investigated the effect of the augmentation methods on the performance of the deep model. We also showed that how end-to-end deep learning models perform better compared with hand-crafted features. But, is the success of the proposed framework all due to the deep learning models? In the next portion of the work, we will look at this question and investigate the significance of different steps of the pipeline in the performance of the CNN-LSTM model.

4.4.6.4 Ablation study on the pipeline

In this work, we utilized three strategies to improve the performance of the deep learning models: data preprocessing, data augmentation, and the SWA. The first two strategies are two of the main steps in the proposed pipeline, and the last strategy is implemented in the

design of the deep learning model. Figure 4.25 shows the average classification accuracy with an ablation study on these strategies. As we observe in this figure, preprocessing has the most impact on the performance of the network. Removing the preprocessing step from the proposed pipeline results in a 20% reduction in the average accuracy and more than 5% increase in the variance. Removing SWA from the deep learning model has the least amount of impact on the average classification score compared with the complete pipeline. However, removing SWA from the model increases the likelihood of generating outlier classification scores. The result with no augmentation is the same as in Figure 4.24.

This analysis shows the importance of the preprocessing step in the performance of the model. One of the major difficulties in using a skeleton model relates to the fact that they are noisy and contain irrelevant information [95]. Based on the results of this experiment, we speculate that applying the proposed preprocessing (removing noisy joints and mid-torso joint centering of the skeleton) has a major impact on alleviating these challenges. In this work, we design deep learning models that take minimally preprocessed skeleton joints as the input. The results in Figure 4.25 indicate that the success of the presented pipeline is not solely based on the design of a deep learning model. The handling and preprocessing of the data play important roles in the success of the model. The average classification accuracy without preprocessing is around 70%, which is even lower than the best result with a non-deep learning model (refer to SVM scores for *selected normalized joints* input in Table 4.16). By comparing the results in Figure 4.25, we also realize that while data augmentation can improve the performance of the model, it is not as effective as the preprocessing step. This observation indicates the importance of proper preprocessing of the skeleton data for a successful pathological gait classification. Next, we will look at the effect of the number of classes on the performance of two models, one deep learning and one non-deep learning. Our goal is to see how

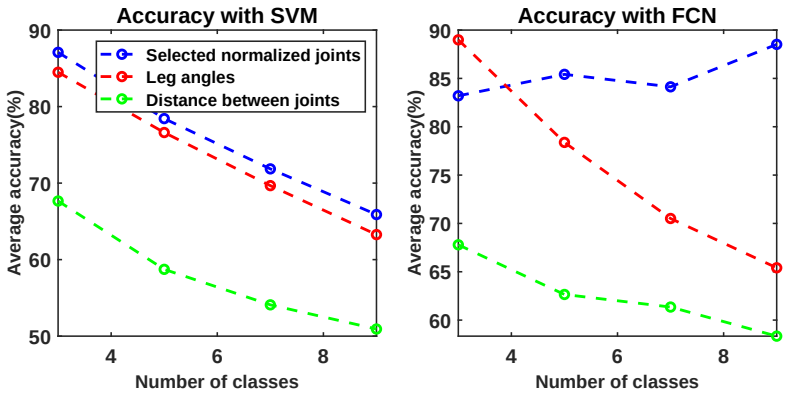


FIGURE 4.26: Average classification accuracy for three features for the different number of gait anomaly classes. The graphs on the left and right side of the figure show the results with the SVM and the FCN, respectively.

resilient each feature is to the number of gait abnormality classes. We are also interested in studying the performance of a deep model when we increase the number of pathological gait classes.

4.4.6.5 Effect of the number of classes

In real-world scenarios, we encounter different types of gait anomalies caused by surgical operations, injuries, and various physiological and neurological disorders. Therefore, it is essential to investigate how the performance of a particular model or set of features changes as the number of abnormality classes varies. To address this challenge, we examine how the performance of different features changes as we increase the number of gait abnormality categories. For this experiment, we use the *Walking gait* dataset. This experiment is carried out with two classifiers: the SVM and the FCN. Both of these classifiers demonstrated an overall best performance over different features (refer to Table 4.16). We use a deep network and a non-neural network classifier to reduce the effect of the classifier and concentrate on the performance of the features instead.

Figure 4.26 presents the average classification accuracy with different numbers of classes. For this experiment, we considered $N_{class} = [3, 5, 7, 9]$, where N_{class} is the set of all the class numbers we used for each experiment. The *Walking gait* dataset has 9 classes

in total. When the number of classes is less than 9, we always select *normal* class as one of the classes, and $N_{class} - 1$ classes are selected at random from the abnormality classes. Except for the case in which $N_{class} = 9$, we carried each experiment 100 times, every time selecting $N_{class} - 1$ number of abnormal classes at random. The results in Figure 4.26 shows the average accuracy over 100 runs of the model. Except for one case, we see that both the SVM and the FCN perform the best when the number of classes is the smallest, $N_{class} = 3$. With the SVM, all features show a linear degradation in the performance as the number of classes increases. With the FCN, we observe a different behavior with selected normalized joints. While the performance of the other two features degrades as the number of classes increases, the FCN shows a more robust performance with a small variance over different numbers of classes. We even observe that the FCN acquires its best performance with selected normalized joints when $N_{class} = 9$. This might be because, for each feature, we use the same model that acquires the best result with the original 9-class dataset. With *leg angles*, we have two noteworthy observations. First, *Leg angles* outperforms other features with $N_{class} = 3$, which might indicate that with smaller numbers of class abnormalities, leg angles can be a more discriminative set of attributes. Second, the performance of the FCN with leg angles as the input degrades with the steepest slope among the comparing features. With distance between joints as the feature, we observe the slowest degradation in performance with both the SVM and the FCN. However, this feature also leads to lower accuracy compared with the other features. As we mentioned before, this might be partly caused by the high dimension of this feature and the requirement for a more complex model to acquire better performance.

4.4.6.6 Experimental results with transfer learning

In this part, we look at the effect of transfer learning, evaluating the performance of the proposed models on two datasets: *MMGS* and *pathological gait*. We use the same

models as was presented in subsection 4.4.3.3, using the weights of the best models to initialize the networks on the target datasets. With the CNN-LSTM model for the selected normalized joints, we change the kernel size in the convolutional layers to 3 for transfer learning. Optimizer changes in a few cases, using either Adam or RMSprop instead of SGD. Other than this, the only parameters of the networks that we change for the target datasets are learning rate, regularization, and batch size. Each of the target models can have one or more of these parameters different from the source model. We also observed that not using SWA in training the deep models on the source dataset (here *Walking gait*) works better for transfer learning compared with applying SWA on the model trained on the source dataset. We speculate that using SWA might make a network too specialized in one dataset and result in overfitting on the source dataset. Therefore, causing the network to undermine some of the more generalizing patterns that can be transferred to another dataset with different classes of abnormalities. Based on this observation, for all the deep learning models, we removed SWA when we trained the model on the source dataset for transfer learning.

In the following paragraphs, we present experimental results for the *MMGS* and *Pathological gait* datasets.

Experimental results with MMGS: Table 4.17 illustrates the mean classification accuracy and F-score for the *MMGS* dataset, using different features as the input of deep learning model. Each deep learning classifier is initialized using the weights of the best corresponding model that was trained on the *Walking gait* dataset. To compensate for the class imbalance in the *MMGS* dataset, we use class weights when fitting the model. The class weights are implemented according to Equation 4.13. For *MMGS* dataset, we also compare the performance of the proposed framework with the work of Khokhlova *et al.* [5]. In the latter study, the authors use low limb flexion angles as the features and design an ensemble of LSTM networks for modeling and classification. It should be noted

TABLE 4.17: Average recognition accuracy and F-score on *MMGS* dataset [5] for the proposed features (selected normalized joints) and other features. Except for the leg flexion feature, all the other results are acquired using transfer learning. It should be noted that the result with the flexion angles is based on an ensemble of five LSTMs. The last three lines show the results when train and validation split is **n**on-**s**ubject-based. The underlined and bold scores present the best results with subject-based and non-subject-based criteria, respectively.

Input	Score	LSTM	FCN	CNN-LSTM
Leg angles [2]	Accuracy	77.10	<u>82.84</u>	81.70
	F-score	75.60	<u>81.57</u>	79.89
Distance between joints [3]	Accuracy	62.20	79.52	71.98
	F-score	61.77	78.58	70.97
Leg flexion angles [5]	Accuracy	82	—	—
	F-score	—	—	—
Selected normalized joints	Accuracy	78.03	80.27	80.74
	F-score	77.14	79.82	80.53
Leg angles [2] (NS)	Accuracy	81.04	83.51	82.55
	F-score	79.44	82.14	80.64
Distance between joints [3] (NS)	Accuracy	63.74	81.35	76.49
	F-score	63.01	80.44	74.96
Selected normalized joints (NS)	Accuracy	**83.64**	81.51	82.71
	F-score	**82.61**	80.31	82.04

TABLE 4.18: Average recognition accuracy and F-score on *MMGS* dataset [5] for the proposed features (selected normalized joints) and other features using three non deep learning classifiers: LR (logistic regression), RF (random forest), and SVM.

Input	Score	LR	RF	SVM
Leg angles [2]	Accuracy	74.56	**78.41**	**81.67**
	F-score	72.13	**76.87**	**80.45**
Distance between joints [3]	Accuracy	74.90	47.49	74.10
	F-score	72.66	47.87	71.97
Leg flexion angles [5]	Accuracy	—	—	77.6
	F-score	—	—	—
Selected normalized joints	Accuracy	**80.08**	60.16	76.09
	F-score	**78.34**	59.45	74.47

that the accuracy reported in Table 4.17 for [5] is based on an ensemble of five LSTM networks. They also reported accuracy based on a single LSTM network that is 77%. Considering the complexity and a large number of parameters in an ensemble of LSTM networks, the LSTM and CNN-LSTM that we designed have less number of parameters, yet can acquire comparable classification scores through transfer learning.

The last three rows of Table 4.17 show the classification scores when train-validation split is **n**on-**s**ubject based. Like before, using a non-subject-based train-validation split criterion, we observe an increase in the classification scores with all the features. The largest improvement is acquired with distance between joints using the CNN-LSTM

model. The results also show that when the subject-based train-validation split is applied (the first four rows of the table), leg angles achieve the best results. With a non-subject-based train-validation split (the last three rows of the table), the proposed feature set demonstrates the best performance.

To have a better understanding of the performance of each feature set, we also compared the three feature vectors with non-deep learning models. In Table 4.18, we present the results of this comparison. The results in this table also show that with the *MMGS* dataset, leg angles outperform other features. Considering the results in Tables 4.17 and 4.18, we observe that leg angles perform better than other features irrespective of the type of classifier for a three-class problem. Here, two points should be considered. First, leg angles build a low dimensional feature vector compared with selected normalized joints and distance between joints. Second, in general, as we increase the dimension of the input vector, we require more data for detecting the relevant discriminatory patterns. Our observations in Figure 4.26 also show that leg angles might be a suitable feature set with a small number of classes. Utilizing transfer learning and data augmentation, we might do better with higher dimensional input features, still, the dataset is relatively small and has a large intra-class variation (a large number of subjects compared with *Walking gait*). These results might suggest that with fewer data and a small number of anomalous classes, we might not need a complicated end-to-end deep learning classifier to acquire high classification scores. Of course, such a statement is limited to our observations as well as the complexity of the target dataset.

Experimental results with Pathological gait: Table 4.19 presents classification accuracy and F-score with the *Pathological gait* dataset, using transfer learning. Classification scores are computed based on the one-vs-all criterion that was employed in the original paper [4]. Every time, the data from one subject is separated for test, and training is performed using the data from the rest of subjects. For each subject selected for

TABLE 4.19: Average recognition accuracy and F-score on *Pathological gait* dataset [4] for the proposed features (selected normalized joints) and other features. All the results are based on transfer learning, except for the result in the last row that is the best reported result in [4]. The results in the last row with LR (logistic regression), RF (random forest), and SVM are based on our experiments, using the leg joints as the feature vector.

Input	Score	LR	RF	SVM	LSTM	FCN	CNN-LSTM	GRU
Leg angles [2]	Accuracy	60.50	83.65	76.78	80.66	83.42	81.55	—
	F-score	61.81	84.39	77.49	81.49	85.08	82.72	—
Distance between joints [3]	Accuracy	80.23	82.74	79.24	81.39	**86.46**	82.87	—
	F-score	82.86	84.74	81.67	82.63	87.75	84.20	—
Selected normalized joints	Accuracy	81.24	**90.83**	77.51	**90.10**	79.63	**89.83**	—
	F-score	81.80	91.45	78.73	90.73	78.73	90.63	—
Leg joints [4]	Accuracy	**87.95**	83.02	**86.88**				**93.67**

test, training and test is performed ten times. The final result is the average over 100 times of training and testing (for which there are 10 subjects in this dataset). The last row of Table 4.19 shows the best result as was reported by Jun *et al.*. They acquired this accuracy by using only the leg joints. For modeling and classification, they designed a gated recurrent unit (GRU) network. The network has five layers, with four GRU layers, and one dense layer, each having 125 units. The network also includes a ReLU layer after the input layer. Other results in the last row is based on our experiments, using the leg joints with LR, RF, and SVM.

The results in Table 4.19 show that leg joints [4] outperform other features. There are couple of points that should be considered here:

- Except for the leg joints case [4], all the results with the deep learning models are reported based on transfer learning. Considering the fact that we designed our models based on another dataset with different classes of abnormalities, the results with the proposed method are fairly reasonable.

- With random forest, we achieved an accuracy of $> \%90$ using selected normalized joints as the feature. None of the other features could acquire such high accuracy with non-deep models. This observation suggests that selected normalized joints is

a discriminatory feature vector that can perform well even without a deep learning model.

- With the *Walking gait* dataset, LSTM performed poorly compared with other deep learning models. However, with transfer learning, LSTM outperformed other deep models with both *MMGS* and *Pathological gait* datasets.

- The FCN model that performed well with selected normalized joints on the *Walking gait* dataset, did poorly on both of the transfer learning Tasks. This might be caused by the high number of parameters in this model (four convolutional layers with 650 units per layer). This observation confirms the fact that complex models cannot generalize well to other datasets. This also suggests that FCN was likely overfitted on the *Walking gait* dataset.

- Compared with other deep learning models in this work, the GRU network in [4] is a computationally expensive model with larger number of parameters (four GRU layers and one dense layer, with 125 units per layer), that also takes longer to train. Large deep learning models that are well tuned to one dataset might not generalize well to unseen data with different and more diverse underlying patterns.

- Comparing the performance of distance between joints and leg angles, we observe that unlike the other two datasets, the former feature set outperforms the latter. This might suggest that given enough data, higher dimensional features can acquire better improvements compared to the low dimensional feature vectors. In particular, this can indicate that designing a model that solely relies on the features that are related to certain joints might not be the best practice. This assumption can be bolstered by considering the fact that gait is a complex pattern that is built upon the interaction between different body parts.

In this dataset, the proposed GRU model by Jun *et al.* that uses leg joints as the input outperforms the other features, including the proposed (selected normalized joints). However, the GRU model was specifically designed for the *Pathological gait dataset*. Also, the versatility of the model was not tested on other datasets. In contrast, our proposed models were designed on another dataset and achieve high classification scores on both the source dataset (*Walking gait*) and two target datasets (*MMGS* and *Pathological gait*), which demonstrates the versatility of the proposed model. Considering the classes of pathological gait anomalies in this dataset that includes antalgic gait, stiff-legged gait, lurching gait, steppage gait, and Trendelenburg gait, we notice these anomalies are mostly related to the locomotion in the lower body. Therefore, it makes sense that leg joints become more relevant and distinctive for these types of gait anomalies. On the contrary, our model looks at more joints that some of which might not be informative for these particular classes of abnormalities, therefore resulting in lower performance. However, the sole purpose of this work is not simply improving classification performance. Gait is a complex physiological system that consists of many subtle interactions between different body muscles. There are many types of gait anomalies that include interactions between lower and upper body joints. For example, in Parkinson's disease, bradykinesia (slowness of movement) can effect both lower and upper body limbs. During tremor which can be observed in various neurological diseases such as Parkinson's disease, cerebral palsy, and Huntington's disease, knee joints, elbows, and shoulder shake heavily. In Ataxia (a degenerative disease that causes impaired coordination), motions of different body parts such as hands, arms, and legs are affected. Therefore, for clinical assessment, we need to observe almost every functional muscle. By considering different body joints, our model is designed to detect distinctive patterns, which are the result of interactions between different body joints. This property makes the proposed pipeline more suitable for adaptation in monitoring symptoms of various ailments that affect gait patterns.

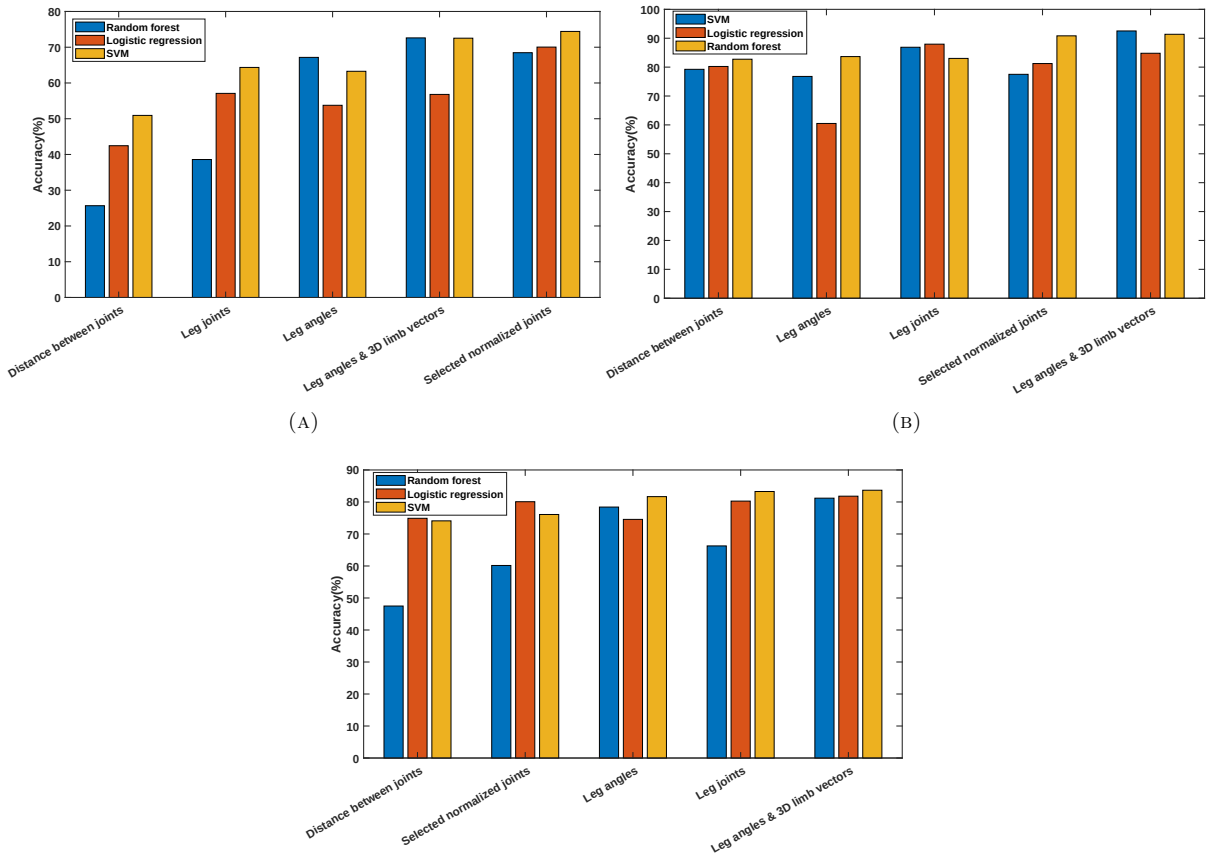


FIGURE 4.27: Average classification accuracy for five types of features: Distance between joints [3], Leg angles [2], Leg joints [4], Selected normalized joints (proposed), Leg angles & 3D limb vectors (proposed) with three classifiers: Logistic regression (LR), Random forest (RF), and Support vector machines (SVM). The results are reported on (A) the *Walking gait* dataset with 9 classes, (B) the *Pathological gait* dataset with 6 classes, and (C) the *MMGS* dataset with 3 classes.

4.4.7 SGAR: Handcrafted vs deep learning-based features

Our analysis and experimental results lead us to two the following questions:

- Among the investigated features, what are the most discriminatory features for SGAR?
- Should a successful SGAR pipeline rely on hand-crafted features or end-to-end deep learning models?

Figure 4.27 presents three graphs, each comparing the performance of five different features: distance between joints [3], leg angles [2], leg joints [4], selected normalized

joints (proposed), leg angles & 3D limb vectors (proposed). The latter feature was introduced in the first SGAR problem that we investigated in this chapter. Each of these graphs shows average classification accuracy for one of the datasets that we employed in this chapter. Comparison between features is performed using three classifiers: logistic regression (LR), random forest (RF), and support vector machines (SVM). We observe that Leg angles & 3D limb vectors outperform the othere features in most of the cases. Selected normalized joints only outperform Leg angles & 3D limb vectors in the *Walking gait* dataset, the dataset with the most number of classes. However, as we see the performance of these two feature sets is quite close in the *Walking gait* dataset. In *MMGS*, selected normalized joints performance comes after three other features. This confirms our previous observation, when the number of classes is small, lower dimensional features perform better. Leg joints [4] that outperformed other competing features in combination with a deep GRU on the *Pathological gait* dataset, does not perform well with the *Walking gait* dataset. Even on the *Pathological gait* dataset, Leg angles & 3D limb vectors and selected normalized joints could acquire accuracy of above 90% with non deep learning models, but Leg joints could only achieve $> 90\%$ accuracy with a computationally expensive GRU model. By comparing the graphs in Figure 4.27, we see the performance of leg joints degrade as the number of classes increase. In contrary, performance of selected normalized joint improves as the number of classes increases.

Our observations in Figure 4.27 suggest that leg angles & 3D limb vector are the most discriminatory features among the competing features. However, we also observe that with the *Walking gait* dataset, selected normalized joints outperforms the latter. What if we have a dataset with more classes? Based on our observations, leg angles & 3D limb vectors and selected normalized joint are both discriminatory features for the SGAR problem. Selected normalized joint also performs quite well with deep learning models and could generalize well to other datasets with different types of gait abnormalities. In

particular, as the number of classes increases, performance of selected normalized joints improves that hints this feature set is likely to perform well with a more diverse set of gait abnormalities.

Leg angles & 3D limb vectors is a handcrafted feature vector. Selected normalized joints is a set of minimally preprocessed skeleton joint coordinates that performed well with both deep learning and non-deep learning models. Our observations suggest that **the design of the features and how they are handled are critical steps in an SGAR framework**. Deep learning models can help to improve the performance of a set of well-designed features. We might need more complex and novel architectures as the number of classes increases. But, relevant and discriminatory features can make this procedure easier. Therefore, a successful SGAR framework can benefit from carefully designed handcrafted features and properly filtered input data in combination with deep learning models to further improve its prediction ability.

4.5 Remarks

In this chapter, we present an efficient pipeline for the classification of different gait abnormalities using the skeleton data. Unlike the prevalent practice of gait anomaly recognition with skeleton data, we investigate this problem in a multi-class framework, designing models for minor gait abnormality recognition. The multi-class layout makes this problem challenging because of the high inter-class similarities. In the first problem, we focus on a small normal/pathological gait dataset, with three classes of normal/pathological gaits. This dataset has a large intra-class variation due to multiple short sequences and a relatively large number of subjects. Many clinical gait assessment methods measure parameters that are extracted from lower body limbs. Therefore, for this problem, we propose a set of features extracted from the lower body skeleton joints. We design a

bidirectional LSTM network for modeling and classification and acquire state-of-the-art accuracy.

Next, we focus on a large dataset with nine classes of normal/pathological gaits. We present a deep learning model that extracts relevant features from the skeleton data. To reduce generalization error, we present an effective data preprocessing and implement proper data augmentation. While deep learning models acquire the best performance in this work, the success of the presented pipeline is mostly due to a proper data preprocessing step. The high prediction accuracy on *Walking gait*, a 9-class normal/pathological gait dataset, demonstrates the superiority of the proposed method. Besides, through transfer learning, we also acquire high classification scores on two other skeleton-based pathological gait datasets that contain different types of gait abnormalities. This further shows the ability of the proposed pipeline to learn the embedded gait patterns that can be generalized to various types of gait abnormalities through transfer learning. The presented pipeline also shows a robust performance with different numbers of class abnormalities that are randomly selected from the *Walking gait* dataset. Our experiments with several features also suggest that while end-to-end deep learning models can improve state-of-the-art performance in SGAR, proper handling of the input data is a significant aspect of a successful SGAR pipeline. **T**his is the first skeleton-based gait anomaly recognition that employs data augmentation and is evaluated on multiple datasets under heterogeneous scenarios. The presented framework can be employed for frequent and convenient gait assessment inside and outside of specialized clinical facilities. Besides, to further improve the performance of a free-living gait assessment system, the presented methodology can be integrated with models that employ other biomedical sensors.

Chapter 5

Conclusion and future work

The study of gait goes back to Aristotle. In *Parts of animals; Movement of animals; Progression of animals/De partibus animalium* [157], Aristotle gives a detailed description of gait in different animals. However, his description was solely based on his observations of locomotion in human beings and various animals [158]. Giovanni Borelli (1608–1679) was the first person who provided a theoretical framework of gait based on his observations from a set of scientifically designed experiments. Wilhelm and Eduard Weber analyzed gait through a set of experimental studies. They published their findings in *Anatomy and mechanics of walking* in 1836, discussing several aspects of gait such as cadence and walking speed, along with positioning of body limbs during motion. In the early 1900s, Eadweard Muybridge in America and Étienne-Jules Marey in France became the first people who studied locomotion through sequences of photographs (credited as the first video analysis in history). Wilhelm Braune and Otto Fischer performed the first 3-dimensional study of gait. Inman and Eberhart started an interdisciplinary group of scientists, physicians, and engineers at the University of California, Berkeley, and made significant contributions to the field of gait analysis. While video cameras and instrumented gait analysis have been around since the 1970s, for years, the application

of designated equipment for the gait analysis has been limited, and mainly cumbersome [158].

Modern gait analysis has been empowered by the advancements in computer hardware, analyzing software, and design of new modalities. Human gait is a complex physiological pattern that conveys important information about the health, gender, age, and identity of individuals. These properties, along with the advancement of new technologies have transformed gait analysis into a major tool for the identification and health evaluation of individuals. In this dissertation, we utilize modalities such as flash lidar and Kinect and provide engineering solutions for the two widespread applications of gait analysis: identification and health evaluation. Both of the presented solutions are model-based, utilizing skeleton information of individuals. The skeleton-related attributes mimic actual physical traits in the human body and provide benefits in terms of data compaction, computation, and scalability.

5.1 Flash lidar-based gait recognition

In the first part of this dissertation (chapter 2 and 3), we present an efficacious pipeline of 3D gait recognition for flash lidar data based on pose estimation and robust correction of erroneous and missing joint measurements. A flash lidar can provide new opportunities for gait recognition through a fast acquisition of depth and intensity data over an extended range of distances. However, the flash lidar data are plagued by artifacts, outliers, noise, and sometimes missing measurements, which negatively affects the performance of existing analytics solutions. At the beginning of this dissertation, we proposed three questions. Two of these questions were related to the first part of this dissertation:

- If the collected data are noisy to a level such that a considerable number of feature vectors contain faulty and missing values, can we still achieve high accuracy and precision in gait recognition?

- When a high percentage of the input features are either noisy or missing, can we avoid data elimination and do any better through model correction?

In the following subsections, we provide a summary of gait recognition with flash lidar data, address these two questions, and outline some avenues for future research.

5.1.1 Summary

By modeling the coordinates of each joint through time as a set of time sequences, we present filtering mechanisms that correct noisy and missing skeleton joint measurements to improve gait recognition. The correction mechanism is valuable because:

- The shortage of data is a major challenge in many real-world surveillance scenarios and data removal will only exacerbate the data scarcity problem. A correcting filtering mechanism can preserve the original data that is costly to collect in numerous applications.

- Data removal will also result in loss of temporal information for cases where corrupted or missing data exists in the successive frames. Temporal information encodes the dynamics of a motion that is critical in describing an activity or identifying an individual.

- Occlusion is a common problem in gait recognition that adversely affects the performance of state-of-the-art methods. When information from a fitted skeleton is used, the presented joint correction filtering can be employed in cases of occlusion. This is because the existence of occlusion would create missing skeleton joints that can be recovered through the joint location correction.

We integrate robust statistics with conventional feature moments to encode the dynamics of the motion. Through a set of experimental results, we show that traditional feature moments can serve as a better representative of motion dynamics after skeleton

correction if they are considered along with robust statistics such as the median, and the lower and upper quartiles. As an alternative method for applications where data elimination is not an issue, we investigate features extracted from noisy skeletons for outliers and present a method for detecting outliers in vector-based features. This contribution is an effort to follow the traditional practice of removing noisy data and performing classification on the remaining higher quality features. With a considerable amount of flash lidar data, outlier removal can be employed. The remaining data can be used in methods with per-frame (one-shot) identification.

Back to the questions that were proposed at the beginning of this section: our results suggest that we can still acquire high classification scores with the flash lidar data that is plagued with a considerable amount of noisy and missing values. Even better, we can achieve higher classification scores when we perform data correction and avoid the common practice of data elimination.

5.1.2 Future work

The presented filtering mechanism improves gait recognition by correcting missing and noisy skeleton joints in two steps. The two step criterion is vital, as it avoids the effect of noisy measurements in generating an initial prediction for the missing values in the first step. A robust smoothing filter in the second step reduces the negative effect of the remaining outliers and noisy prediction of the first step. However, as we have shown in Figure 3.8, there are cases that correction fails. The filtering procedure can fail if a joint or a whole skeleton is missing over multiple consecutive frames. As the number of successive frames with a missing joint or skeleton increases, the failure cases are more likely to happen. With the first-order polynomial fitting in the first step, GlidarPoly loses adequate support as the number of missing skeleton or missing joints increases. On the other hand, higher-order polynomial in GlidarCo over-smooths the final estimation, resulting

in more false predictions. For a better imputation of missing values and correction of noisy measurements, modeling the dynamics of the motion is also helpful. In particular, this becomes essential in more realistic scenarios, where the dynamic of the motion, such as walking speed, might change. For such a direction of study, a larger collection of data from each subject is required.

The flash lidar dataset contains sequences of 10 subjects, who are walking in three different manners that cover poses from multiple views. A major direction for future work calls for a dataset that consists of a larger population of subjects with a more diverse group of settings. This can be beneficial in multiple ways. First, it will open an avenue for training a deep pose estimation tool that generally requires a large and diverse collection of images. In the first step of the presented pipeline, we utilize OpenPose, a state-of-the-art pose estimation tool. Such deep learning-based pose estimators are trained with images collected by optical cameras. Therefore, their performance is adversely affected by the noisy imaging process of flash lidar camera. With a large collection of flash lidar images, flash lidar-based deep pose models can be designed that can improve the performance of skeleton detection. Second, the availability of a large collection of flash lidar data paves the path for a well-designed optimization model to find relevant, yet interpretable features. In this work, we opt for anthropometric-based features to avoid the interpretability issue of a complex feature design. However, with a large data collection, there might be a need for a more distinct set of features to recognize a larger collection of the population considering the limitations of flash lidar modality.

5.2 Multi-class SGAR

The second part of this dissertation (chapter 4) is focused on the application of gait for a health evaluation, providing modeling and classification approaches for minor gait abnormalities. There is an extensive list of ailments that affect gait function, with each having multiple symptoms. Every disease or gait-affecting injury has its own set of

the diagnostic list. There are also overlaps in symptoms between different pathological abnormalities [159]. Therefore, clinical diagnosis involves a long list of lab tests, gait performance assessments, and diagnostic imaging screening. These analyses are costly, time-consuming, and in some cases, inconvenient for the patient. Traditional gait assessment techniques are certainly informative. But, they are mainly limited to specialized labs. These techniques are not a proper representative of the gait patterns that are mostly observed in a free-living environment (collecting data in an uncontrolled and unsupervised environment) [82], yet are vital for an accurate gait assessment.

The last question that we proposed at the beginning of this dissertation concerned the application of gait anomaly recognition:

- Can we perform frequent gait evaluation without the need for an equipped specialized lab, in the convenience of living environment with a low-cost and markerless equipment such as Kinect?

In the following subsections, we provide a summary for the multi-class SGAR that was presented in this dissertation, address the above question, and outline future work proposals.

5.2.1 Summary

We employ the data collected by Kinect for a contact-free, markerless approach of minor pathological gait classification. Sequences are simulated by healthy subjects and there are multiple recordings of the same abnormality imitation by each subject. Besides, subjects do not walk at the same speed. These facts will result in intra-class dissimilarity and inter-class similarities, which makes minor pathological gait recognition quite challenging. We build a multi-class and computationally-efficient framework that can be adapted for convenient out-of-the-lab gait evaluation. Our contributions are three-fold:

- Unlike the majority of the SGAR studies (and gait anomaly recognition in general), **this work presents a multi-class framework**. The multi-class framework makes this study adaptable for real-world medical diagnosis of gait abnormalities, where each class can represent a minor gait anomaly or a specific symptom.

- We employ data augmentation that is vital for reducing the generalization error in deep learning models. The significance of data augmentation for applications such as SGAR with limited data becomes even more significant. Unlike computer vision tasks, data augmentation for time series-based problems is not trivial and depends on the nature of the dataset. To the best of our knowledge, this is the **first study that employs data augmentation as a part of the SGAR pipeline**.

- The majority of state-of-the-art SGAR studies focus on a single dataset and are not capable of extending to a broad range of input data due to the mismatch between simple handcrafted features and the complexity of the human gait. **For the first time**, we present a model that is evaluated on the three largest publicly available skeleton-based gait anomaly datasets. Since each of these datasets represent different types of anomalies, **design of the model is not dependent on the specific patterns in one dataset** and can be adopted for similar tasks. In contrast to standard SGAR solutions, we apply transfer learning to two other datasets and provide an evaluation.

For the multi-class pathological gait classification, we design multiple deep learning-based models, both for the proposed features and competing methods. The design of proper deep learning models is an engineering problem that requires an understanding of deep networks' structure and the underlying concepts such as gradient-based learning methods, overfitting and underfitting of the models, regularization techniques, data augmentation, and transfer learning. However, designing sophisticated deep learning models

for the SGAR problem is not the purpose of this study. Our main goal is to develop a pipeline for SGAR that is effective, computationally efficient, can achieve high prediction accuracy, and can be adapted to datasets with different types of anomalies. We believe, the presented framework provides a proper solution for gait anomaly recognition using skeleton data. This work is also focused to set forth a model that is resilient to the number of investigated classes. Our experimental results with transfer learning indicate that the presented research can easily be extended to recognize other types of gait anomalies out of the designated lab facilities. Numerous age-related ailments manifest themselves by gait pattern alternation. Frequent and convenient monitoring of the elderly population can be helpful by diagnosing such early symptoms. The early diagnosis of age-related diseases can improve life quality, extend life expectancy, and reduce treatment and diagnostic costs. The presented SGAR mechanism can be combined with models that employ other modalities such as wearable sensors to improve the prediction capability of a free-living gait evaluation framework. The results of this study present an affirmative response to the proposed question, indicate the potential of markerless equipment such as Kinect for a convenient and frequent out of the lab minor gait anomaly classification.

Our results suggest that designing relevant features and the proper handling of these features are an important aspect of a successful SGAR pipeline. Such features can be integrated with the well-designed deep networks to accomplish state-of-the-art performance for pathological gait identification.

5.2.2 Future work

The gait assessment methodologies in clinics are the results of decades of studies. These methods are based on parameters that are interpretable and highly accurate. However, such gait assessment methodologies require specialized labs and involve a cumbersome

procedure for patients and healthcare providers. These evaluations are costly and time-consuming. A question that naturally arises is: How can we create models and methods that make gait assessment affordable and convenient that can be performed in a doctor's office, assisted living environment, or the convenience of home, with minimum to none supervision?

To address this question and considering the result of the current study, here we outline a few proposals for future research.

Our observations show that certain handcrafted features can be valuable in the SGAR problem. Numerous studies in the past have shown that combining deep learning-based attributes with non-deep learning features can result in state-of-the-art performance in different fields. One avenue for future study can be to design models to fuse deep learning-based features with proper handcrafted attributes to acquire multi-class SGAR frameworks with higher prediction accuracy. With handcrafted attributes, we integrate domain knowledge and interpretability to the model. With deep learning-based features, we introduce more abstract attributes that are designed based on maximum class separation.

Our results indicate that while non-deep learning models could achieve high classification scores in a few cases, deep networks outperformed other models in the majority of scenarios. A major challenge in gait anomaly recognition is limited data availability. In general, a large dataset is one of the main requirements for acquiring high performance with a deep learning model. While data collection for various tasks in computer vision and natural language processing is not an issue, there are applications for which data collection is quite costly and time-consuming. Designing a proper augmentation method is valuable for applications such as skeleton-based normal/pathological gait recognition due to data limitation, difficulties of data collection, and confidentiality concerns for patients. In this work, we used data augmentation for skeleton-based gait anomaly recognition and achieved improvement in the classification of gait abnormalities. A future study can

concentrate on adopting and evaluating other augmentation methods from relevant tasks. Besides, considering the nature of the SGAR problem, new augmentation methods should be designed and examined.

Finally, with limited publicly available data for SGAR, a systematic study of deep learning and non-deep learning methods for a large population with a more diverse set of abnormalities is missing. Such study is critical in recognizing the most discriminative features in this field and designing proper generative models for clinical and non-clinical applications.

We live in an era with fast-paced progress in advanced hardware and software design, and healthcare is one of the fields with many underlying potentials to benefit from such advancement. In this dissertation, we aimed to address one of the problems in this field and tried to pave the path for future studies that take a multi-class approach toward the problem of gait recognition with the goal of real-world applications.

Publications resulting from this thesis

Journal Publications

1. **Nasrin Sadeghzadehyazdi**; Tamal Batabyal; Alexander Glandon; Nibir Dhar; Babajide Familoni; Khan Iftekharuddin; Scott Acton, "Using skeleton correction to improve flash lidar-based gait recognition", Journal of visual Communication and Image Representation (JVCI) (Under review)

2. Iftekharuddin, Khan; Glandon, Alexander; Vidyaratne, Lasitha ; Dhar, Nibir; **Sadeghzadehyazdi, Nasrin**; Familoni, Jide ; Acton, Scott T., "View Invariant 3D Skeleton Estimation and Human Identification Using Lidar Full Motion Video", IEEE transactions on Systems, Man, and Cybernetics: Systems (Under review)

3. **Nasrin Sadeghzadehyazdi**; Tamal Batabyal; Scott Acton, "Modeling spatiotemporal patterns of gait anomaly with a CNN-LSTM deep neural network", Journal of Information Science (In preparation)

Conference Publications

1. **Sadeghzadehyazdi, Nasrin**; Batabyal, Tamal; Glandon, Alexander; Dhar, Nibir K.; Familoni, BO; Iftekharuddin, Khan M; Acton, Scott T., "Glidar3DJ: A View-Invariant gait identification via flash lidar data correction" 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019.

2. Glandon, Alexander and Vidyaratne, Lasitha and **Sadeghzadehyazdi, Nasrin**; Dhar, Nibir K; Familoni, Jide O; Acton, Scott T; and Iftekharuddin, Khan M., "3D Skeleton Estimation and Human Identity Recognition Using Lidar Full Motion Video" 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, 2019.

3. Wang, Jie; Fu, Zhongxiao; **Sadeghzadehyazdi, Nasrin**; Kipnis, Jonathan; Acton, Scott T., "Nonlinear shape regression for filtering segmentation results from calcium imaging" 2018 25th IEEE International Conference on Image Processing (ICIP). IEEE, 2018.

4. **Sadeghzadehyazdi, Nasrin**; Batabyal, Tamal; Barnes, Laura E.; Acton, Scott T., "Graph-based classification of healthcare provider activity" 2016 50th Asilomar Conference on Signals, Systems and Computers. IEEE, 2016.

Bibliography

[1] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[2] T.-N. Nguyen, H.-H. Huynh, and J. Meunier, "Skeleton-based abnormal gait detection," *Sensors*, vol. 16, no. 11, p. 1792, 2016.

[3] M. Meng, H. Drira, M. Daoudi, and J. Boonaert, "Detection of abnormal gait from skeleton data," 2016.

[4] K. Jun, Y. Lee, S. Lee, D.-W. Lee, and M. S. Kim, "Pathological gait classification using kinect v2 and gated recurrent neural networks," *IEEE Access*, vol. 8, pp. 139 881–139 891, 2020.

[5] M. Khokhlova, C. Migniot, A. Morozov, O. Sushkova, and A. Dipanda, "Normal and pathological gait classification lstm model," *Artificial intelligence in medicine*, vol. 94, pp. 54–66, 2019.

[6] J. E. Cutting and L. T. Kozlowski, "Recognizing friends by their walk: Gait perception without familiarity cues," *Bulletin of the psychonomic society*, vol. 9, no. 5, pp. 353–356, 1977.

[7] C. P. Charalambous, "Walking patterns of normal men," in *Classic Papers in Orthopaedics*. Springer, 2014, pp. 393–395.

[8] J. D. Eggleston, J. R. Harry, R. A. Hickman, and J. S. Dufek, "Analysis of gait symmetry during over-ground walking in children with autism spectrum disorder," *Gait & Posture*, vol. 55, pp. 162–166, 2017.

[9] I. Bouchrika, M. Goffredo, J. Carter, and M. Nixon, "On using gait in forensic biometrics," *Journal of forensic sciences*, vol. 56, no. 4, pp. 882–889, 2011.

[10] I. Bouchrika, J. N. Carter, and M. S. Nixon, "Towards automated visual surveillance using gait for identity recognition and tracking across multiple non-intersecting cameras," *Multimedia Tools and Applications*, vol. 75, no. 2, pp. 1201–1221, 2016.

[11] J. Han and B. Bhanu, "Individual recognition using gait energy image," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 2, pp. 316–322, 2005.

[12] A. K. Jain, R. Bolle, and S. Pankanti, *Biometrics: personal identification in networked society*. Springer Science & Business Media, 2006, vol. 479.

[13] D. Kastaniotis, I. Theodorakopoulos, C. Theoharatos, G. Economou, and S. Fotopoulos, "A framework for gait-based recognition using kinect," *Pattern Recognition Letters*, vol. 68, pp. 327–335, 2015.

[14] K. Shiraga, Y. Makihara, D. Muramatsu, T. Echigo, and Y. Yagi, "Geinet: View-invariant gait recognition using a convolutional neural network," in *2016 international conference on biometrics (ICB)*. IEEE, 2016, pp. 1–8.

[15] Y. He, J. Zhang, H. Shan, and L. Wang, "Multi-task gans for view-specific feature learning in gait recognition," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 102–113, 2018.

[16] J. Lu and Y.-P. Tan, "Gait-based human age estimation," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 761–770, 2010.

[17] S. Yu, T. Tan, K. Huang, K. Jia, and X. Wu, "A study on gait-based gender classification," *IEEE Transactions on image processing*, vol. 18, no. 8, pp. 1905–1910, 2009.

[18] Z. A. Khan and W. Sohn, "Abnormal human activity recognition system based on r-transform and kernel discriminant technique for elderly home care," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1843–1850, 2011.

[19] C. Prakash, R. Kumar, and N. Mittal, "Recent developments in human gait research: parameters, approaches, applications, machine learning techniques, datasets and challenges," *Artificial Intelligence Review*, vol. 49, no. 1, pp. 1–40, 2018.

[20] W. Chi, J. Wang, and M. Q.-H. Meng, "A gait recognition method for human following in service robots," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1429–1440, 2018.

[21] S. Del Din, A. Godfrey, and L. Rochester, "Validation of an accelerometer to quantify a comprehensive battery of gait characteristics in healthy older adults and parkinson's disease: toward clinical and at home use," *IEEE journal of biomedical and health informatics*, vol. 20, no. 3, pp. 838–847, 2016.

[22] O. Ťupa, A. Procházka, O. Vyšata, M. Schätz, J. Mareš, M. Vališ, and V. Mařík, "Motion tracking and gait feature estimation for recognising parkinson's disease using ms kinect," *Biomedical engineering online*, vol. 14, no. 1, p. 97, 2015.

[23] V. J. Verlinden, J. N. van der Geest, A. Hofman, and M. A. Ikram, "Cognition and gait show a distinct pattern of association in the general population," *Alzheimer's & Dementia*, vol. 10, no. 3, pp. 328–335, 2014.

[24] A. Mannini, D. Trojaniello, A. Cereatti, and A. Sabatini, "A machine learning framework for gait classification using inertial sensors: Application to elderly, post-stroke and huntington's disease patients," *Sensors*, vol. 16, no. 1, p. 134, 2016.

[25] A. Ball, D. Rye, F. Ramos, and M. Velonaki, "Unsupervised clustering of people from 'skeleton'data," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2012, pp. 225–226.

[26] M. Kumar and R. V. Babu, "Human gait recognition using depth camera: a covariance based approach," in *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*. ACM, 2012, p. 20.

[27] K. Yang, Y. Dou, S. Lv, F. Zhang, and Q. Lv, "Relative distance features for gait recognition with kinect," *Journal of Visual Communication and Image Representation*, vol. 39, pp. 209–217, 2016.

[28] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1611.08050*, 2016.

[29] R. Alp Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7297–7306.

[30] H. Lakany, "Extracting a diagnostic gait signature," *Pattern recognition*, vol. 41, no. 5, pp. 1627–1637, 2008.

[31] S. R. Simon, "Quantification of human motion: gait analysis—benefits and limitations to its application to clinical problems," *Journal of biomechanics*, vol. 37, no. 12, pp. 1869–1880, 2004.

[32] K. Kaczmarczyk, A. Wit, M. Krawczyk, and J. Zaborski, "Gait classification in post-stroke patients using artificial neural networks," *Gait & posture*, vol. 30, no. 2, pp. 207–210, 2009.

[33] M. R. Daliri, "Automatic diagnosis of neuro-degenerative diseases using gait dynamics," *Measurement*, vol. 45, no. 7, pp. 1729–1734, 2012.

[34] A. Leardini, G. Lullini, S. Giannini, L. Berti, M. Ortolani, and P. Caravaggi, "Validation of the angular measurements of a new inertial-measurement-unit based rehabilitation system: comparison with state-of-the-art gait analysis," *Journal of neuroengineering and rehabilitation*, vol. 11, no. 1, pp. 1–7, 2014.

[35] R. Baker, "Gait analysis methods in rehabilitation," *Journal of neuroengineering and rehabilitation*, vol. 3, no. 1, p. 4, 2006.

[36] K. Lorincz, B.-r. Chen, G. W. Challen, A. R. Chowdhury, S. Patel, P. Bonato, M. Welsh *et al.*, "Mercury: a wearable sensor network platform for high-fidelity motion analysis." in *SenSys*, vol. 9, 2009, pp. 183–196.

[37] J. C. Schlachetzki, J. Barth, F. Marxreiter, J. Gossler, Z. Kohl, S. Reinfelder, H. Gassner, K. Aminian, B. M. Eskofier, J. Winkler *et al.*, "Wearable sensors objectively measure gait parameters in parkinson's disease," *PloS one*, vol. 12, no. 10, p. e0183989, 2017.

[38] M. Gabel, R. Gilad-Bachrach, E. Renshaw, and A. Schuster, "Full body gait analysis with kinect," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2012, pp. 1964–1967.

[39] H. Mousavi Hondori and M. Khademi, "A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation," *Journal of medical engineering*, vol. 2014, 2014.

[40] K. Otte, B. Kayser, S. Mansow-Model, J. Verrel, F. Paul, A. U. Brandt, and T. Schmitz-Hübsch, "Accuracy and reliability of the kinect version 2 for clinical measurement of motor function," *PloS one*, vol. 11, no. 11, p. e0166532, 2016.

[41] A. Kale, A. Sundaresan, A. Rajagopalan, N. P. Cuntoor, A. K. Roy-Chowdhury, V. Kruger, and R. Chellappa, "Identification of humans using gait," *IEEE Transactions on image processing*, vol. 13, no. 9, pp. 1163–1173, 2004.

[42] C. Benedek, "3d people surveillance on range data sequences of a rotating lidar," *Pattern Recognition Letters*, vol. 50, pp. 149–158, 2014.

[43] M. Hofmann, S. M. Schmidt, A. N. Rajagopalan, and G. Rigoll, "Combined face and gait recognition using alpha matte preprocessing," in *2012 5th IAPR International Conference on Biometrics (ICB)*. IEEE, 2012, pp. 390–395.

[44] D. S. Matovski, M. S. Nixon, S. Mahmoodi, and J. N. Carter, "The effect of time on gait recognition performance," *IEEE transactions on information forensics and security*, vol. 7, no. 2, pp. 543–552, 2011.

[45] H. Fujiyoshi, A. J. Lipton, and T. Kanade, "Real-time human motion analysis by image skeletonization," *IEICE TRANSACTIONS on Information and Systems*, vol. 87, no. 1, pp. 113–120, 2004.

[46] A. F. Bobick and A. Y. Johnson, "Gait recognition using static, activity-specific parameters," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[47] B. Gálai and C. Benedek, "Feature selection for lidar-based gait recognition," in *Computational Intelligence for Multimedia Understanding (IWCIM), 2015 International Workshop on*. IEEE, 2015, pp. 1–5.

[48] Y. Iwashita, K. Uchino, and R. Kurazume, "Gait-based person identification robust to changes in appearance," *Sensors*, vol. 13, no. 6, pp. 7884–7901, 2013.

[49] J. Tang, J. Luo, T. Tjahjadi, and F. Guo, "Robust arbitrary-view gait recognition based on 3d partial similarity matching," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 7–22, 2016.

[50] M. Babaee, L. Li, and G. Rigoll, "Gait recognition from incomplete gait cycle," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 768–772.

[51] P. Chattopadhyay, S. Sural, and J. Mukherjee, "Frontal gait recognition from incomplete sequences using rgb-d camera," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 11, pp. 1843–1856, 2014.

[52] T. Batabyal, A. Vaccari, and S. T. Acton, "Ugrasp: A unified framework for activity recognition and person identification using graph signal processing," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3270–3274.

[53] T. Batabyal, S. T. Acton, and A. Vaccari, "Ugrad: A graph-theoretic framework for classification of activity with complementary graph boundary detection," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1339–1343.

[54] R. A. Clark, K. J. Bower, B. F. Mentiplay, K. Paterson, and Y.-H. Pua, "Concurrent validity of the microsoft kinect for assessment of spatiotemporal gait variables," *Journal of biomechanics*, vol. 46, no. 15, pp. 2722–2725, 2013.

[55] R. M. Araujo, G. Graña, and V. Andersson, "Towards skeleton biometric identification using the microsoft kinect sensor," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013, pp. 21–26.

[56] J. Preis, M. Kessel, M. Werner, and C. Linnhoff-Popien, "Gait recognition with kinect," in *1st international workshop on kinect in pervasive computing*. New Castle, UK, 2012, pp. 1–4.

[57] A. Sinha, K. Chakravarty, and B. Bhowmick, "Person identification using skeleton information from kinect," in *Proc. Intl. Conf. on Advances in Computer-Human Interactions*, 2013, pp. 101–108.

[58] F. Ahmed, P. P. Paul, and M. L. Gavrilova, "Dtw-based kernel and rank-level fusion for 3d gait recognition using kinect," *The visual computer*, vol. 31, no. 6-8, pp. 915–924, 2015.

[59] S. Ali, Z. Wu, X. Li, N. Saeed, D. Wang, and M. Zhou, "Applying geometric function on sensors 3d gait data for human identification," in *Transactions on Computational Science XXVI*. Springer, 2016, pp. 125–141.

[60] R. Horaud, M. Hansard, G. Evangelidis, and C. Ménier, "An overview of depth cameras and range scanners based on time-of-flight technologies," *Machine vision and applications*, vol. 27, no. 7, pp. 1005–1020, 2016.

[61] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Y. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, 2015, pp. 388–394.

[62] S. Zennaro, "Evaluation of microsoft kinect 360 and microsoft kinect one for robotics and computer vision applications," 2014.

[63] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 816–833.

[64] V. B. Semwal, J. Singha, P. K. Sharma, A. Chauhan, and B. Behera, "An optimized feature selection technique based on incremental feature analysis for bio-metric gait data classification," *Multimedia tools and applications*, vol. 76, no. 22, pp. 24 457–24 475, 2017.

[65] T. P. Hettmansperger and J. W. McKean, *Robust nonparametric statistical methods*. CRC Press, 2010.

[66] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation," *SIAM Journal on Numerical Analysis*, vol. 17, no. 2, pp. 238–246, 1980.

[67] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *Journal of the American statistical association*, vol. 74, no. 368, pp. 829–836, 1979.

[68] M. Smolik, V. Skala, and O. Nedved, "A comparative study of lowess and rbf approximations for visualization," in *International Conference on Computational Science and Its Applications*. Springer, 2016, pp. 405–419.

[69] B. Dikovski, G. Madjarov, and D. Gjorgjevikj, "Evaluation of different feature sets for gait recognition using skeletal data from kinect," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2014, pp. 1304–1308.

[70] S. Han, "The influence of walking speed on gait patterns during upslope walking," *Journal of Medical Imaging and Health Informatics*, vol. 5, no. 1, pp. 89–92, 2015.

[71] J. Kovač and P. Peer, "Human skeleton model based dynamic features for walking speed invariant gait recognition," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[72] K. Koide and J. Miura, "Identification of a specific person using color, height, and gait features for a person following robot," *Robotics and Autonomous Systems*, vol. 84, pp. 76–87, 2016.

[73] M. Munaro, A. Basso, A. Fossati, L. Van Gool, and E. Menegatti, "3d reconstruction of freely moving persons for re-identification with a depth sensor," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4512–4519.

[74] O. Oreifej, R. Mehran, and M. Shah, "Human identity recognition in aerial images," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 709–716.

[75] A. Wu, W.-S. Zheng, and J.-H. Lai, "Robust depth-based person re-identification," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2588–2603, 2017.

[76] Y. Zhang and S. Li, "Gabor-lbp based region covariance descriptor for person re-identification," in *2011 Sixth International Conference on Image and Graphics*. IEEE, 2011, pp. 368–371.

[77] S. Liao, Y. Hu, X. Zhu, and S. Z. Li, "Person re-identification by local maximal occurrence representation and metric learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2197–2206.

[78] M. Munaro, A. Fossati, A. Basso, E. Menegatti, and L. Van Gool, "One-shot person re-identification with a consumer depth camera," in *Person Re-Identification*. Springer, 2014, pp. 161–181.

[79] A. Haque, A. Alahi, and L. Fei-Fei, "Recurrent attention models for depth-based person identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1229–1238.

[80] N. Sadeghzadehyazdi, T. Batabyal, A. Glandon, N. K. Dhar, B. Familoni, K. Iftekharuddin, and S. T. Acton, "Glidar3dj: A view-invariant gait identification via flash lidar data correction," *arXiv preprint arXiv:1905.00943*, 2019.

[81] J.-Y. Kwon, H. J. Chang, J. Y. Lee, Y. Ha, P. K. Lee, and Y.-H. Kim, "Effects of hippotherapy on gait parameters in children with bilateral spastic cerebral palsy," *Archives of physical medicine and rehabilitation*, vol. 92, no. 5, pp. 774–779, 2011.

[82] N. Takayanagi, M. Sudo, Y. Yamashiro, S. Lee, Y. Kobayashi, Y. Niki, and H. Shimada, "Relationship between daily and in-laboratory gait speed among healthy community-dwelling older adults," *Scientific reports*, vol. 9, no. 1, pp. 1–6, 2019.

[83] S. Del Din, A. Godfrey, B. Galna, S. Lord, and L. Rochester, "Free-living gait characteristics in ageing and parkinson's disease: impact of environment and ambulatory bout length," *Journal of neuroengineering and rehabilitation*, vol. 13, no. 1, p. 46, 2016.

[84] R. Morris, A. Hickey, S. Del Din, A. Godfrey, S. Lord, and L. Rochester, "A model of free-living gait: A factor analysis in parkinson's disease," *Gait & posture*, vol. 52, pp. 68–71, 2017.

[85] R. D. Gurchiek, R. H. Choquette, B. D. Beynnon, J. R. Slauterbeck, T. W. Tourville, M. J. Toth, and R. S. McGinnis, "Open-source remote gait analysis: a post-surgery patient monitoring application," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[86] K. K. Patterson, N. K. Nadkarni, S. E. Black, and W. E. McIlroy, "Gait symmetry and velocity differ in their relationship to age," *Gait & posture*, vol. 35, no. 4, pp. 590–594, 2012.

[87] M. R. Daliri, "Chi-square distance kernel of the gaits for the diagnosis of parkinson's disease," *Biomedical Signal Processing and Control*, vol. 8, no. 1, pp. 66–70, 2013.

[88] Y. Wu and L. Shi, "Analysis of altered gait cycle duration in amyotrophic lateral sclerosis based on nonparametric probability density function estimation," *Medical engineering & physics*, vol. 33, no. 3, pp. 347–355, 2011.

[89] E. Baratin, L. Sugavaneswaran, K. Umapathy, C. Ioana, and S. Krishnan, "Wavelet-based characterization of gait signal for neurological abnormalities," *Gait & posture*, vol. 41, no. 2, pp. 634–639, 2015.

[90] Q. Ye, Y. Xia, and Z. Yao, "Classification of gait patterns in patients with neurodegenerative disease using adaptive neuro-fuzzy inference system," *Computational and mathematical methods in medicine*, vol. 2018, 2018.

[91] V. Dietz, K. Fouad, and C. Bastiaanse, "Neuronal coordination of arm and leg movements during human locomotion," *European Journal of Neuroscience*, vol. 14, no. 11, pp. 1906–1914, 2001.

[92] A. Paiement, L. Tao, S. Hannuna, M. Camplani, D. Damen, and M. Mirmehdi, "Online quality assessment of human movement from skeleton data," in *British Machine Vision Conference*. BMVA press, 2014, pp. 153–166.

[93] Q. Li, Y. Wang, A. Sharf, Y. Cao, C. Tu, B. Chen, and S. Yu, "Classification of gait anomalies from kinect," *The Visual Computer*, vol. 34, no. 2, pp. 229–241, 2018.

[94] A. A. Chaaraoui, J. R. Padilla-López, and F. Flórez-Revuelta, "Abnormal gait detection with rgb-d devices using joint motion history features," in *2015 11th IEEE international conference and workshops on automatic face and gesture recognition (FG)*, vol. 7. IEEE, 2015, pp. 1–6.

[95] K. Jun, D.-W. Lee, K. Lee, S. Lee, and M. S. Kim, "Feature extraction using an rnn autoencoder for skeleton-based abnormal gait recognition," *IEEE Access*, vol. 8, pp. 19 196–19 207, 2020.

[96] L. Tao, A. Paiement, D. Damen, M. Mirmehdi, S. Hannuna, M. Camplani, T. Burghardt, and I. Craddock, "A comparative study of pose representation and dynamics modelling for online motion quality assessment," *Computer vision and image understanding*, vol. 148, pp. 136–152, 2016.

[97] X. S. Papageorgiou, G. Chalvatzaki, C. S. Tzafestas, and P. Maragos, "Hidden markov modeling of human pathological gait using laser range finder for an assisted living intelligent robotic walker," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6342–6347.

[98] S. Potluri, S. Ravuri, C. Diedrich, and L. Schega, "Deep learning based gait abnormality detection using wearable sensor system," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2019, pp. 3613–3619.

[99] A. Zhao, L. Qi, J. Li, J. Dong, and H. Yu, "A hybrid spatio-temporal model for detection and severity rating of parkinson's disease from gait data," *Neurocomputing*, vol. 315, pp. 1–8, 2018.

[100] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[101] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[102] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," in *Advances in neural information processing systems*, 2013, pp. 351–359.

[103] R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Advances in neural information processing systems*, 2001, pp. 402–408.

[104] L. Palmerini, L. Rocchi, S. Mazilu, E. Gazit, J. M. Hausdorff, and L. Chiari, "Identification of characteristic motor patterns preceding freezing of gait in parkinson's disease using wearable sensors," *Frontiers in neurology*, vol. 8, p. 394, 2017.

[105] J. Nonnekes, R. J. Goselink, E. Ruuvzivcka, A. Fasano, J. G. Nutt, and B. R. Bloem, "Neurological disorders of gait, balance and posture: a sign-based approach," *Nature Reviews Neurology*, vol. 14, no. 3, p. 183, 2018.

[106] D. Phan, N. Nguyen, P. N. Pathirana, M. Horne, L. Power, and D. Szmulewicz, "A random forest approach for quantifying gait ataxia with truncal and peripheral measurements using multiple wearable sensors," *IEEE Sensors Journal*, vol. 20, no. 2, pp. 723–734, 2019.

[107] F. Mortazavi and A. Nadian-Ghomsheh, "Stability of kinect for range of motion analysis in static stretching exercises," *PloS one*, vol. 13, no. 7, p. e0200992, 2018.

[108] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy, "Evaluation of pose tracking accuracy in the first and second generations of microsoft kinect," in *Healthcare Informatics (ICHI), 2015 International Conference on.* IEEE, 2015, pp. 380–389.

[109] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv preprint arXiv:1603.06995*, 2016.

[110] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," 2016.

[111] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[112] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[113] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[114] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[115] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2015, pp. 707–712.

[116] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *International Conference on Artificial Neural Networks.* Springer, 2005, pp. 799–804.

[117] A. Zeyer, R. Schlüter, and H. Ney, "Towards online-recognition with deep bidirectional lstm acoustic models." in *Interspeech*, 2016, pp. 3424–3428.

[118] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," *arXiv preprint arXiv:1801.02143*, 2018.

[119] H. He, G. Huang, and Y. Yuan, "Asymmetric valleys: Beyond sharp and flat local minima," in *Advances in Neural Information Processing Systems*, 2019, pp. 2553–2564.

[120] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[121] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, "A simple baseline for bayesian uncertainty in deep learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 153–13 164.

[122] B. Athiwaratkun, M. Finzi, P. Izmailov, and A. G. Wilson, "There are many consistent explanations of unlabeled data: Why you should average," *arXiv preprint arXiv:1806.05594*, 2018.

[123] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," *arXiv preprint arXiv:2002.08791*, 2020.

[124] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[125] Y. LeCun, I. Kanter, and S. A. Solla, "Second order properties of error surfaces: Learning time and generalization," in *Advances in neural information processing systems*, 1991, pp. 918–924.

[126] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[127] O. Steven Eyobu and D. S. Han, "Feature representation and data augmentation for human activity classification based on wearable imu sensor data using a deep lstm neural network," *Sensors*, vol. 18, no. 9, p. 2892, 2018.

[128] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[129] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*, 2017, pp. 933–941.

[130] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 9–14.

[131] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 565–571.

[132] C. V. Dung *et al.*, "Autonomous concrete crack detection using deep fully convolutional neural network," *Automation in Construction*, vol. 99, pp. 52–58, 2019.

[133] P. Mobadersany, S. Yousefi, M. Amgad, D. A. Gutman, J. S. Barnholtz-Sloan, J. E. V. Vega, D. J. Brat, and L. A. Cooper, "Predicting cancer outcomes from

histology and genomics using convolutional networks," *Proceedings of the National Academy of Sciences*, vol. 115, no. 13, pp. E2970–E2979, 2018.

[134] F. F. Ting, Y. J. Tan, and K. S. Sim, "Convolutional neural network improvement for breast cancer classification," *Expert Systems with Applications*, vol. 120, pp. 103–115, 2019.

[135] A. Dutta, T. Batabyal, M. Basu, and S. T. Acton, "An efficient convolutional neural network for coronary heart disease prediction," *Expert Systems with Applications*, p. 113408, 2020.

[136] T. Hosaka, "Bankruptcy prediction using imaged financial ratios and convolutional neural networks," *Expert systems with applications*, vol. 117, pp. 287–299, 2019.

[137] S. Kiranyaz, T. Ince, R. Hamila, and M. Gabbouj, "Convolutional neural networks for patient-specific ecg classification," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015, pp. 2608–2611.

[138] O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman, "Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks," *Journal of Sound and Vibration*, vol. 388, pp. 154–170, 2017.

[139] D. George, X. Xie, and G. K. Tam, "3d mesh segmentation via multi-branch 1d convolutional neural networks," *Graphical Models*, vol. 96, pp. 1–10, 2018.

[140] I. El Maachi, G.-A. Bilodeau, and W. Bouachir, "Deep 1d-convnet for accurate parkinson disease detection and severity prediction from gait," *Expert Systems with Applications*, vol. 143, p. 113075, 2020.

[141] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[142] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[143] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[144] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[145] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[146] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *International conference on machine learning*. PMLR, 2017, pp. 2208–2217.

[147] A. Khatami, M. Babaie, H. R. Tizhoosh, A. Khosravi, T. Nguyen, and S. Nahavandi, "A sequential search-space shrinking using cnn transfer learning and a radon projection pool for medical image retrieval," *Expert Systems with Applications*, vol. 100, pp. 224–233, 2018.

[148] S. Lu, Z. Lu, and Y.-D. Zhang, "Pathological brain detection based on alexnet and transfer learning," *Journal of computational science*, vol. 30, pp. 41–47, 2019.

[149] S. Chaabouni, J. Benois-Pineau, and C. B. Amar, "Transfer learning with deep networks for saliency prediction in natural video," in *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1604–1608.

[150] X. Yuan, D. Li, D. Mohapatra, and M. Elhoseny, "Automatic removal of complex shadows from indoor videos using transfer learning and dynamic thresholding," *Computers & Electrical Engineering*, vol. 70, pp. 813–825, 2018.

[151] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Transfer learning for time series classification," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1367–1376.

[152] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[153] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[154] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[155] T.-N. Nguyen and J. Meunier, "Walking gait dataset: point clouds, skeletons and silhouettes," *DIRO, University of Montreal, Tech. Rep.*, p. 1379, 2018.

[156] H. Alshazly, C. Linse, E. Barth, and T. Martinetz, "Handcrafted versus cnn features for ear recognition," *Symmetry*, vol. 11, no. 12, p. 1493, 2019.

[157] A. L. Peck, E. S. Forster *et al.*, "Parts of animals; movement of animals; progression of animals/de partibus animalium." Harvard University Press;, Tech. Rep., 1937.

[158] R. Baker, "The history of gait analysis before the advent of modern computers," *Gait & posture*, vol. 26, no. 3, pp. 331–342, 2007.

[159] G. P. Bella, N. B. Rodrigues, P. J. Valenciano, L. M. Silva, and R. C. Souza, "Correlation among the visual gait assessment scale, edinburgh visual gait scale and observational gait scale in children with spastic diplegic cerebral palsy," *Revista Brasileira de Fisioterapia*, vol. 16, no. 2, pp. 134–140, 2012.