HAW HAMBURG

BACHELOR THESIS
Thore Ottenheym

Graph Convolutional Network based Gesture Interpretation of Sekeleton Data

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Thore Ottenheym

Graph Convolutional Network based Gesture Interpretation of Sekeleton Data

Bachelor thesis submitted for examination in Bachelor´s degree
in the study course *Bachelor of Science Angewandte Informatik*
at the Department Computer Science
at the Faculty of Engineering and Computer Science
at University of Applied Science Hamburg

Supervisor: Prof. Dr. Kai von Luck
Supervisor: Jan Schwarzer, Ph.D.

Submitted on: 13.09.2024

Thore Ottenheym

Thema der Arbeit

Graph Convolutional Network based Gesture Interpretation of Sekeleton Data

Stichworte

Skelettdaten, Graph Convolution Network, Gesture Interpretation

Kurzzusammenfassung

Diese thesis untersucht die Anwendbarkeit von Graph Convolutional Networks (GCNs) zur Automatisierung der Interpretation von in-the-wild Skelettdaten. Die Studie untersucht, ob GCNs, insbesondere das Spatial-Temporal Graph Convolutional Network (ST-GCN), Gesten, die in einer unkontrollierten Umgebung aufgenommen wurden, effizient erkennen können. Es wurde ein Datensatz mit Daten aus dem HopE-Projekt erstellt und eine Trainingspipeline entwickelt, die Techniken des Transfer learnings und der Data Augmentation beinhaltet. Die Ergebnisse zeigen, dass GCNs in der Lage sind, die räumliche und zeitliche Dynamik zu erfassen, die für eine genaue Gestenerkennung in realen Szenarien notwendig ist, und geben einen Einblick in das Potenzial von GCNs zur Verbesserung der automatischen Gesteninterpretation in unterschiedlichen und unvorhersehbaren Umgebungen.

Thore Ottenheym

Title of Thesis

Graph Convolutional Network based Gesture Interpretation of Sekeleton Data

Keywords

Skeleton data, Graph Convolution Network, Gesture Interpertation

Abstract

This thesis investigates the applicability of Graph Convolutional Networks (GCNs) for automating the interpretation of in-the-wild skeletal data. The study examines whether

GCNs, specifically the Spatial-Temporal Graph Convolutional Network (ST-GCN), can effectively interpret gestures captured in uncontrolled environments. A dataset was created using data from the HopE project, and a training pipeline was developed that incorporates transfer learning and data augmentation techniques. The results demonstrate that GCNs can capture the spatial and temporal dynamics necessary for accurate gesture recognition in real-world scenarios, and provide insight into the potential of GCNs to improve automated gesture interpretation in diverse and unpredictable environments.

Contents

List of Figures

List of Tables

List of Acronyms

HAW Hochschule für Angewandte Wissenschaften

LSTM Long short-term memory

RNN Recurrent Neural Network

CNN Convolutional Neural Network

GCN Graph Convolutional Network

ST-GCN Spatial Temporal Graph Convolutional Network

FoV Field of View

RGB Red Green Blue

DTW Dynamic Time Warping

GAN Generative Adversarial Network

1 Introduction

In recent years, many organizations have adopted hybrid work models that require employees to be on-site a certain number of days per week. This approach seeks to balance the flexibility of remote working with the benefits of face-to-face collaboration and social interaction [9]. In this context, understanding the dynamics of workplace interactions has become increasingly important, especially in hybrid work environments where informal communication plays an important role in maintaining team cohesion[9].

As part of the HopE[1] project, an ambient display equipped with a body-tracking system was installed in the work café environment of a software development company. The system tracks the body movement of individuals as they interact with the display and each other. While the primary goal of the project is to understand the honeypot effect, the data collected also provides valuable insights into the behavior of developers when they are on-site.

One frequent observation from the data is that many individuals are often seen holding a coffee cup. This gesture, particularly when multiple people are involved, could suggest informal communication. Specific gesture and action patterns provide different cues: a person crossing the room while holding a coffee may indicate they are transitioning into or out of a social interaction, while a person sitting and drinking coffee could suggest they are actively engaged in an informal interaction, especially when others are present. These subtle behaviors may offer insights into how informal communication unfolds and its role in fostering team dynamics within a hybrid work environment.

In addition, the data collected by the HopE project spans a long period of time, making it possible to analyze long-term patterns in employee behavior. This is important for understanding how workplace interactions evolve over time, especially in a hybrid work environment where employee presence can vary from day to day. To effectively interpret this vast amount of data, automation techniques are essential.

[1] https://csti.haw-hamburg.de/project/HopE/(Accessed 2024-07-31)

Graph Convolutional Networks (GCNs) have shown promise in gesture and action recognition by effectively modeling the spatial and temporal relationships in skeleton data [36]. GCNs extend traditional convolutional networks to non-Euclidean data, making them well suited for tasks involving complex human motion. Their ability to capture dependencies across joints in the body allows for more accurate gesture interpretation, even in unstructured and dynamic environments such as those observed in the HopE project. Given their strengths, GCNs offer a compelling approach for automating the interpretation of gestures and behaviors captured in real-world settings, where conditions are less controlled and more varied than in laboratory environments.

1.1 Research Aim

The goal of this thesis is to investigate whether Graph Convolutional Networks (GCNs), specifically the Spatial-Temporal Graph Convolutional Network (ST-GCN), can effectively detect and interpret specific gestures and behaviors from the in-the-wild body-tracking data collected by the HopE project.

To achieve this goal, the thesis will first review existing approaches for handling skeleton data and managing limited training datasets, including the use of transfer learning and data augmentation. Based on these findings, a pipeline will be developed to preprocess the data, create a dataset, and train the ST-GCN model. The performance of the model will be evaluated to identify the best methods for accurate gesture recognition of in-the-wild data.Ultimately, this research aims to support the automation of pattern recognition and gesture interpretation, with the goal of enhancing the understanding of employee behavior.

1.2 Thesis Structure

This thesis is divided into five chapters. Following the introduction, Chapter 2 presents the context of the acquired data and reviews the relevant literature on gesture recognition from skeleton data and graph convolutional networks, providing a comprehensive overview of existing methodologies and identifying gaps that this thesis aims to address. Chapter 3 details the methodology used, including the process of dataset creation, the design, and training of the neural network model, and the experiments used to evaluate

the model's performance. Chapter 4 presents the results of the experiments and discusses the implications. Finally, Chapter 5 summarizes the main findings of the thesis and suggests directions for future research, with the goal of contributing to the field of skeleton data analysis and providing insights into employee behavior in hybrid work environments.

2 Analysis

This chapter begins by introducing the dataset and its collection context, followed by a discussion of preprocessing techniques used to enhance skeleton data. It then delves into data augmentation and transfer learning strategies before presenting an exploration of Graph Convolutional Networks (GCNs) and their application to gesture interpretation. In Section 2.1, the context of the dataset used in this thesis is introduced, including details on the environment in which the data was collected and the HopE project from which it originates. Section 2.2 focuses on preprocessing techniques for skeleton data, particularly methods to remove noise and improve data quality, such as jitter reduction and filtering techniques. Section 2.3 presents various data augmentation methods and their application to skeleton-based data to improve model generalization and performance. Section 2.4 introduces Graph Convolutional Networks (GCNs), with a special focus on the Spatial-Temporal Graph Convolutional Network (ST-GCN) model, detailing its technical features and relevance to gesture interpretation tasks. Section 2.5 provides an introduction to the NTU-RGB+D dataset. In Section 2.6, transfer learning techniques are discussed, outlining how they are applied to improve model performance in gesture recognition tasks. Finally, Section 2.7 presents the problem statement, highlighting the challenges of gesture interpretation in real-world environments and formulating the central research question of this thesis.

2.1 Context of the Dataset

This section begins with an overview of human representation in spatiotemporal space, covering different approaches such as local feature-based and skeleton-based methods. This is followed by an overview of the HopE project, including details on the environment in which the data were collected, the equipment used, and the data format. Understanding these elements is essential to address the challenges of processing and analyzing the captured skeleton data.

2.1.1 Human Representation in Spatiotemporal Space

The problem of human representation in spatio-temporal space is a well-studied area in computer vision. The methods used can be broadly categorized into two main approaches: local feature-based representations and skeleton-based representations[13].

Local feature-based methods focus on detecting key points of interest within the spatiotemporal dimensions. These points are characterized using patches centered on them, which are then encoded into a representation (e.g., using bag-of-word models). While effective at capturing localized details, these methods often overlook the spatial relationships between features[13]. This limitation can hinder the accurate representation of multiple individuals within the same scene, as the connections between features are not clearly identified.

In contrast, skeleton-based representations provide a more structured approach to human representation. As shown by Johansson et al. [15], even a minimal set of tracked joints can effectively represent human behavior. The skeleton represents the joint structure of the human body, not in a biological sense, but rather as a model consisting of a predefined number of joint points tracked over time. Such data can be captured in both 2D and 3D formats, depending on the method used. The quality and accuracy of the skeleton data is influenced by the chosen acquisition method, which also determines the number of joint points tracked.

Approaches to skeleton data acquisition can be categorized into four primary methods:

- **RGB-Based Approaches:** These approaches use cameras to capture RGB images or video data, from which algorithms extract skeleton models. In some cases, cameras equipped with additional sensors can provide depth information, enabling the creation of 3D skeleton models. However, these methods typically require significant storage due to the need to store large amounts of video data. In addition, privacy concerns arise when these methods are used in public spaces.
- **Machine Learning Approaches on Video Data:** These methods rely on machine learning algorithms to process video data and extract 2D skeleton models. Unlike previous methods, these approaches typically do not incorporate depth information, resulting in 2D representations of human motion. While these methods are beneficial in scenarios where 3D data is not required, they are limited by the lack of depth information, which can reduce the accuracy of spatial representations.

- **Marker-Based Systems:** In marker-based systems, markers are placed on the subject's body and tracked using infrared or other techniques. These systems are highly accurate, but are generally limited to laboratory environments due to the complexity of the setup and the controlled conditions required.
- **Depth Camera-Based Approaches:** Depth cameras produce depth images from which skeleton models are generated. This method has the advantage of requiring less storage space than video-based approaches, and also offers greater privacy since it does not capture detailed visual information about the subject. However, the accuracy depends on the hardware used and is prone to occlusion. In addition, subjects must be within a certain range to be captured.

In summary, human representation in spatio-temporal space can be approached through local feature-based or skeleton-based methods. While local feature methods capture detailed points, they may miss important spatial relationships. Skeleton-based methods, especially those that use depth cameras to generate 3D models, provide a more structured and practical solution.

2.1.2 HopE Project

The HopE[1] project is a collaborative research initiative between the HAW Hamburg and the University of the Bundeswehr Munich that aims to understand human behavior in the presence of ambient displays in public and semi-public spaces. The project focuses specifically on the honeypot effect and distinguishes it from related phenomena such as the novelty effect [20].

Another goal of the project is to develop systematic approaches to evaluate the collected data, thereby reducing the reliance on manual analysis [29]. To facilitate data visualization, the University of the Bundeswehr Munich developed the tool PoseViz[2] and the .pose format, which will be described in detail in section 2.4.

This thesis builds on the research and data collected in the HopE project.

[1] https://csti.haw-hamburg.de/project/HopE/(Accessed 2024-07-31)
[2] https://poseviz.com/(Accessed 2024-07-31)

2.1.3 Setup and Environment for Data Collection

In July 2023, the project initiated a collaboration with Körber Pharma[3], an agile software development company. As part of this partnership, a Microsoft Kinect camera was installed in a work café environment. Due to privacy concerns, it was essential to avoid capturing identifiable video data, which precluded methods involving video-based pose estimation. Instead, the Kinect was chosen for its ability to collect anonymous skeleton data. The camera was positioned over an ambient display that initially displayed pharmaceutical and intranet news, along with an architectural map. Over time, the display's functionality was expanded to include interactive activities, such as playing games against each other.

The work café is a semi-open space where people move freely, creating a dynamic environment ideal for data collection. Figure 2.1 illustrates the room setup, including the placement of the Kinect camera and the display, with the arrow indicating the camera's position. The camera's field of view (FoV) captures key areas such as the counter, the first table, and the surrounding room.

The room features an open kitchen-like space with a refrigerator, sink, and coffee maker, as well as various seating options. This setup encourages a range of activities, including grabbing a drink or coffee, meeting with colleagues, or working at one of the seating areas. Analysis of the captured data revealed that the Kinect primarily documented general behavior and interactions within the space, as direct interaction with the display was minimal.

2.1.4 Data Format and Limitations

The dataset captured at the Körber Pharma[4] site spans from July 2023 to March 2024 and includes 5,451 time series with lengths ranging from a few seconds to over 60 minutes. These unfiltered files are stored on a local server in the `.pose` format, with format specifications provided in Figure A.1 and Figure A.2. Each file begins with metadata, including tilt angle and field of view. This is followed by the captured skeleton data, starting with a timestamp. The data is organized into frames, denoted as f, numbered according to the milliseconds elapsed since the start of the recording.

[3] https://www.koerber-pharma.com/(Accessed 2024-07-31)
[4] https://www.koerber-pharma.com/(Accessed 2024-07-31)

Figure 2.1: Design of the Work Café environment with the arrow indicating the position of the Kinect v2 camera

Copyright: Körber Pharma Software

Each frame may contain multiple people, each identified by a unique person ID, represented by p followed by a number. The person ID is followed by the x, y, and z coordinates representing the orientation of the individual, although these are not relevant to this work. The tracked joint points are then listed, marked by a k followed by a number between 0 and 24. This number serves as an identifier for the joint point, as shown in the figure 2.2. For each joint, the coordinates x, y, and z indicate the location of the joint.

While the file format contains additional information, only the elements described above are required for the purposes of this paper.

Despite its practicality and affordability, the Microsoft Kinect camera has limitations in the accuracy of the data it collects. The camera is most accurate when subjects are positioned within a depth range of 0.5 m to 3 m from the sensor, and within 0.5 m to the left or right of the sensor center [39]. As shown in Figure 2.3, accuracy decreases significantly outside of this range, posing a challenge for accurate data collection. To address these limitations, preprocessing techniques can be used to improve the quality and usability of skeleton data.

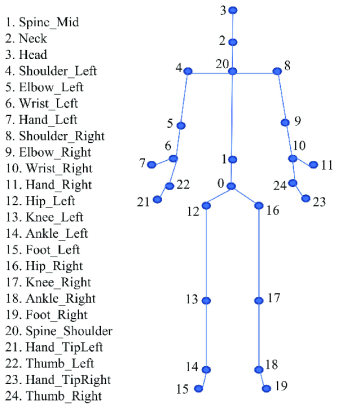


Figure 2.2: Skeleton joints provided by Kinect v2

Source: Keskes et al. [18]

Copyright IEEE 2021

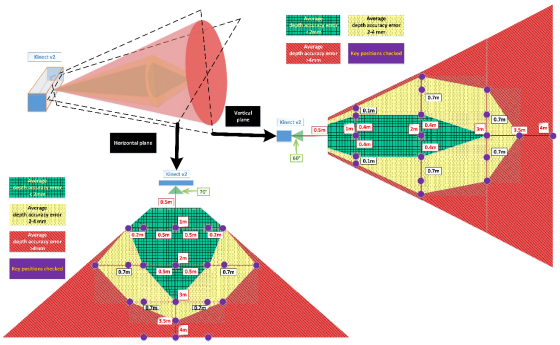


Figure 2.3: Kinect Accuracy

Source: Yang et al. [39]

Copyright IEEE 2015

2.2 Gesture Interpretation of In-the-Wild Skeleton Data

Gesture interpretation involves recognizing and understanding human gestures, which are physical movements or expressions typically involving the hands, arms, face, or body. Unlike traditional machine learning tasks that process data frame by frame, gesture interpretation requires sequential analysis of movements. This is because gestures are dynamic and can only be accurately identified by considering a sequence of frames in context, rather than individual frames in isolation. Approaches to gesture interpretation

have evolved to include convolutional neural networks (CNNs), recurrent neural networks (RNNs), graph convolutional networks (GCNs), and transformers [36].

However, despite advances in these machine learning techniques, most research has been conducted on datasets such as NTU RGB+D [30], which are captured in controlled laboratory environments. These environments ensure high-quality data where gestures are well-defined, making it easier for models to achieve strong performance. In contrast, gesture interpretation in the wild remains less explored [12]. In-the-wild data, collected in uncontrolled environments, presents significant challenges due to its unpredictability and lower quality. Unlike laboratory data, where actions can be directed and controlled, in-the-wild data involves observing actions as they naturally occur.

To address this gap in research, Gupta et al. [12] introduced two new datasets: Skeletics-152 and Skeleton-Mimetics. These datasets were specifically designed to benchmark in-the-wild skeleton action recognition, capturing actions in environments with high variability and contextual differences. Their study revealed a notable performance gap when state-of-the-art models such as MS-G3D [24] and 4s-ShiftGCN [6] were applied to these in-the-wild datasets, compared to their performance in controlled settings. This highlights the need for models that can better handle the complexity of in-the-wild data.

2.3 Preprocessing Techniques

To improve data quality and reduce noise in skeleton data, several preprocessing approaches have been investigated. These techniques operate specifically at the signal level, which focuses on the raw data itself, rather than at the feature level, where feature selection and extraction take place.

A commonly used technique is the application of low-pass filters, such as the Butterworth filter, which smoothes the data and reduces noise [16, 7, 11, 26].

The Butterworth filter is used in various orders and cutoff frequencies tailored to specific applications. Kan et al. [16] used a sixth-order Butterworth filter with a cutoff frequency of 2 Hz in their human action recognition system, demonstrating significant noise reduction and improved data quality. Cogollor et al. [7] used a first-order Butterworth filter with a cutoff frequency of 5 Hz to track manual tasks performed by patients with apraxia and action disorganization syndrome, effectively smoothing the data for better analysis. Faity et al. [11] used a second-order Butterworth filter with a cutoff frequency of 2.5

Hz to assess the validity and reliability of Kinect v2 for analyzing upper limb reaching kinematics, ensuring accurate and consistent kinematic data.

In addition to the Butterworth filter, other filtering methods have been evaluated for their effectiveness in preprocessing Kinect data. Mobini et al. [26] conducted a comparative study of four filtering techniques: Reduced Moving Average, Butterworth, B-Spline, and Kalman filters. Their research showed that the B-Spline filter outperformed the others in their specific context, which involved analyzing acceleration and velocity data captured by the Kinect sensor. In addition, they found that a fourth-order Butterworth filter with a cutoff frequency of 3 Hz struck an optimal balance between noise reduction and data fidelity, making it suitable for their application.

2.4 Data Augmentation

Data augmentation is an important technique in machine learning that aims to expand the size and diversity of training datasets by generating synthetic data from existing examples.[5] This is especially important in domains where collecting and labeling large datasets is challenging, making data augmentation an essential tool for improving model performance.

The approach to data augmentation can vary depending on the nature of the data. For models that process data on a frame-by-frame basis, such as convolutional neural networks (CNNs), augmentation is typically applied independently to each frame. This involves transformations such as rotations, translations, and noise injection, which help to create new variations of the data and improve the model's ability to generalize[37].

When dealing with sequential data, such as video sequences or time series data, augmentation must be applied consistently to all frames in a sequence. This ensures that temporal correlations are preserved, since inconsistent augmentation within a sequence can break continuity and make the data unusable.

In the context of skeleton data, augmentation techniques are generally categorized into spatial, temporal, and generative approaches[37].

[5] While data augmentation can improve the robustness and generalization of machine learning models, it also carries the risk of creating a reality gap. The reality gap refers to the discrepancies between the augmented data and the real-world data, which can lead to reduced model performance in real-world applications. This gap occurs because the synthetic transformations may not fully capture the complexities and variations present in real-world scenarios[33].

Spatial augmentation involves geometric transformations such as translation, rotation, scaling, and shearing. These changes alter the spatial configuration of the data, creating different variations that help models generalize across different poses and perspectives. For example, Wang et al.[35] applied rotation, scaling, and shearing transformations to 3D skeleton data, while Li et al.[21] introduced random rotation and Gaussian noise to increase variability.

Temporal augmentation focuses on enhancing the temporal dynamics of the data by applying techniques such as Gaussian blur, time reversal, interpolation, shifting, and warping. These methods reduce noise, smooth motion trajectories, and adjust sequence lengths, allowing models to better capture the dynamics of human motion over time[37].

Generative Adversarial Networks (GANs), are used to create entirely new data sequences, providing a powerful approach to data augmentation. Notable studies by Shen et al.[31] and Meng et al.[25] have shown that incorporating GAN-generated data can improve the accuracy of Long Short-Term Memory (LSTM) networks. Similarly, Park et al.[27] introduced an LSTM autoencoder network for data augmentation, which further improved the performance of recurrent neural networks (RNNs) by generating more diverse training examples[27, 25].

Despite these advances, GANs still face challenges. Eggert et al. [10] pointed out that GANs often suffer from problems such as mode collapse and lack of variance, resulting in synthetic images that are too similar. This limitation hinders the effectiveness of GANs in generating truly diverse datasets.

For real-world applications, such as surveillance systems, skeleton data often contain imperfections such as missing or noisy points. To address this, Cormier et al.[8] developed a data augmentation framework that introduces random occlusions, such as setting entire frames or specific body parts (e.g., a leg) to zero, to simulate real-world conditions. This approach, also used by Chen et al.[5], helps models deal with incomplete or noisy data. In addition, techniques such as interpolation and keypoint swapping or mirroring were implemented, resulting in a 5% performance improvement on the UAVHuman dataset[23].

2.5 Introduction to the NTU-RGB+D Dataset

The NTU-RGB+D[6] dataset, introduced by Shahroudy et al. [30], is a benchmark for evaluating skeleton-based action recognition models. In addition, it is used for transfer learning, which further emphasizes its importance for improving model generalization across tasks.

The dataset includes 60 different actions, such as walking, waving, drinking, and jumping, performed by 40 different subjects. The data was captured using three time-of-flight (ToF) cameras, specifically Microsoft Kinect v2 cameras, from multiple viewpoints.

The dataset includes two analysis protocols: cross-subject and cross-view. The cross-subject protocol divides the data by different subjects for training and testing, and evaluates model generalization across individuals. The cross-view protocol divides the data by camera angle and evaluates the view-invariance of the model.

With 56,880 samples, the NTU-RGB+D dataset is one of the largest and most diverse for action recognition research, providing a testbed for model development and benchmarking [1]. This diversity, combined with the challenges introduced by the Kinect v2 cameras, such as noise and jitter, make it a challenging dataset for the field of action recognition.

2.6 Graph Convolutional Networks (GCNs)

Graph Convolutional Networks (GCNs) have emerged as a highly effective approach to gesture recognition, especially with skeleton data. They are well suited to the non-Euclidean, graph-like structure of skeleton data, where joints are represented as nodes and their connections as edges [19]. The main advantages of GCNs include their natural alignment with the graph structure of skeleton data, allowing direct processing of joint connections without the need for complex transformations [1]. In addition, GCNs can effectively capture both spatial relationships between joints and temporal dynamics across frames, often combining the strengths of both CNNs and RNNs [1]. Recent studies have demonstrated the effectiveness of GCNs in gesture recognition tasks, achieving high accuracy and robustness in various scenarios [1].

[6] Officially available at https://rose1.ntu.edu.sg/dataset/actionRecognition/ (Accessed 2024-07-31)

Ahmad et al. [1] introduce a new taxonomy for GCN-based action recognition, classifying the methods into five categories:

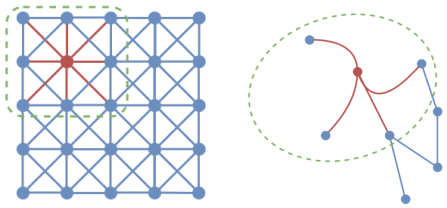
1. **Spatio-temporal GCN:** These models capture both spatial and temporal features of the skeleton data. Examples include ST-GCN [38] and its variants.
2. **Recurrent-attention GCN:** These models integrate attention mechanisms and recurrent neural networks to focus on important joints and time steps. An example is the graph attention network introduced by Veliković et al. [34].
3. **Two-multistream GCN:** These models process multiple streams of data, such as joint coordinates and bone vectors, to enhance feature extraction. The 2s-AGCN [32] is a notable example.
4. **Encoder-decoder GCN:** These models use an encoder-decoder architecture to reconstruct or predict future frames of skeleton data. Examples include the work by Li et al. [22].
5. **Miscellaneous GCN:** This category includes models that do not fit neatly into the other categories but still leverage graph convolutional techniques for action recognition.

Given the advantages of GCNs, they are chosen as the machine learning approach for this thesis. Specifically, the ST-GCN model is selected due to its proven effectiveness in capturing both spatial and temporal dynamics of skeleton data[38]. The ST-GCN model achieved good results on the NTU-RGB+D dataset, with an accuracy of 81.5% in the cross-subject mode and 88.3% in the cross-view mode, highlighting its ability to effectively handle both subject variability and viewpoint changes [38].

2.6.1 Introduction to Graph Convolutional Networks

Graph Convolutional Networks (GCNs) were first introduced by Kipf and Welling [19]. As the name suggests, GCNs perform convolutions on graphs in much the same way that Convolutional Neural Networks (CNNs) perform convolutions on images. This analogy is illustrated in Figure 2.4, where the left image shows traditional Euclidean convolutions and the right image shows non-Euclidean convolutions.

A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E. An edge $e_{i,j} = (v_i, v_j)$ connects a node v_i to a node v_j. The structure of a graph is typically



(a) Euclidean Convolution (b) Non-Euclidean Convolution

Figure 2.4: Comparison of Euclidean and Non-Euclidean Convolutions

Source: Bhatti et al. [3]

represented by an $N \times N$ adjacency matrix A, where $A_{ij} = 1$ if there is an edge between nodes i and j, and 0 otherwise. Each node v_i is associated with a feature vector x_i, which can contain various attributes such as the coordinates of skeleton data.

The degree matrix D is a diagonal matrix where $D_{ii} = \sum_j A_{ij}$. The feature matrix $X \in \mathbb{R}^{N \times d}$ contains the feature vectors for all N nodes, where d is the number of features per node.

A GCN operates on a graph G by stacking multiple graph convolution layers that transform the feature vectors of each node. Each layer takes as input the node vectors from the previous layer and outputs new vectors for each node, analogous to the operation of a CNN layer. This transformation involves aggregating information from a node's neighbors, as shown in Figure 2.4.

Mathematically, the function for a graph convolution layer is given by

$$f(X, A) = \sigma(D^{-1/2}(A + I)D^{-1/2}XW)$$

Here, $A + I$ denotes the adjacency matrix with self-loops added (where I is the identity matrix), ensuring that each node includes its own feature in the aggregation. The terms $D^{-1/2}$ are used to normalize the adjacency matrix. The matrix multiplication $D^{-1/2}(A + I)D^{-1/2}$ normalizes and symmetrizes the adjacency matrix. The feature matrix X is then multiplied by this normalized adjacency matrix and then by the weight matrix W learned

during training. The activation function σ is applied element by element to introduce nonlinearity.

The resulting matrix H consists of the transformed (or embedded) feature vectors for each node. This process allows the GCN to effectively aggregate and propagate information across the graph, enabling it to learn representations that capture the underlying structure and features of the graph data.

2.6.2 Spatial-Temporal Graph Convolutional Network (ST-GCN)

ST-GCN extends the GCN framework to handle temporal dynamics in addition to spatial relations. In ST-GCN, a spatio-temporal graph $G = (V, E)$ is constructed, where V is the set of all joints in the sequence and E includes intra-body and inter-frame links. The feature vector $F(v_{ti})$ consists of joint coordinates and confidence values.

Formally, the edge set E consists of:

- E_S:
$$E_S = \{(v_{t_i}, v_{t_j}) \mid (i, j) \in H\} \tag{2.1}$$

- E_F:
$$E_F = \{(v_{t_i}, v_{(t+1)_i})\} \tag{2.2}$$

To model spatiotemporal dynamics, the concept of neighborhood is extended to include temporally connected joints:

$$B(v_{ti}) = \{v_{qj} \mid d(v_{tj}, v_{ti}) \leq K, |q - t| \leq \Gamma/2\}.$$

Here Γ controls the temporal range and is called the temporal kernel size.

The label map for a spatiotemporal neighborhood rooted at v_{ti} is

$$l_{\mathrm{ST}}(v_{qj}) = l_{ti}(v_{tj}) + (q - t + \Gamma/2) \times K,$$

where $l_{ti}(v_{tj})$ is the label map for the single frame case at v_{ti}.

This formulation allows for a well-defined convolution operation on the constructed spatio-temporal graphs, effectively integrating both spatial and temporal information.

2.7 Transfer Learning

Transfer learning is a widely used technique in machine learning that is particularly effective when dealing with limited datasets. It uses the knowledge gained from pre-training a model on a large, well-labeled dataset and then fine-tuning it on a smaller, task-specific dataset. By transferring the learned features from the source domain to the target domain, transfer learning helps mitigate the problems of overfitting and poor generalization that often occur with small datasets, thereby improving model performance[14].

As discussed by Hosna et al. [14], transfer learning can be categorized into different strategies, including inductive, transductive, and unsupervised transfer learning, each serving different types of tasks and domain relationships. The authors highlight its broad applicability, from real-world simulations and games to medical imaging and sentiment analysis. In particular, the technique's ability to adapt models from one domain to another has proven essential for improving performance in diverse applications, including image classification and recommendation systems. However, challenges such as sample selection bias and negative transfer remain critical to the effective application of transfer learning.

In a specific application, Keskes et al. [18] proposed an approach using a Spatial-Temporal Graph Convolutional Network (ST-GCN) pre-trained on the NTU-RGB+D dataset [30]. This large dataset provided a solid base of learned spatial and temporal features, as described in Section 2.3. Keskes et al. [18] then fine-tuned the pre-trained ST-GCN model using the FallFree dataset, which is specifically designed for fall detection [2]. This strategy allowed the model to adapt to the unique characteristics of fall detection while retaining the general features learned from the NTU-RGB+D dataset. Among the various strategies they explored, one particular transfer learning approach achieved 100% accuracy in their experiments, demonstrating the effectiveness of this technique.

2.8 Problem Statement

Building on the findings of Ahmad et al. [1], it is clear that gesture interpretation of skeleton data has been extensively researched in controlled laboratory environments. However, these studies do not address the challenges of applying machine learning models to in-the-wild skeleton data, especially in real-world work environments such as those

captured by the HopE project. The existing literature highlights the gap in research focused on real-world scenarios [12].

To bridge this gap, this work aims to investigate whether graph convolutional networks (GCNs), specifically the spatial-temporal graph convolutional network (ST-GCN) [38], can help automate gesture interpretation of skeleton data in the wild.

A labeled dataset is created from the data collected by the HopE project, which consists of 5,451 time series. To improve data quality and reduce noise, preprocessing techniques will be applied. In addition, dynamic time warping will be used to identify similar samples within the dataset. The dataset will be classified based on specific gestures observed in the working café environment.

The ST-GCN model will be trained using different strategies, including transfer learning and data augmentation, to improve its performance. The effectiveness of these preprocessing and training methods will be evaluated to identify the optimal approach for gesture recognition in this real-world setting.

Therefore, the central research question of this thesis is

> *Can Graph Convolutional Networks (GCNs), in particular the ST-GCN model, help to automate gesture interpretation in a real-world environment using in-the-wild skeleton data?*

This thesis is based on data and findings from the HopE project, which investigates human behavior in relation to ambient displays in public and semi-public spaces. The results are intended to contribute to a broader understanding and application of machine learning techniques in real-world scenarios.

3 Methodology

This chapter presents the methodology used to evaluate the effectiveness of the Spatial-Temporal Graph Convolutional Network (ST-GCN) for detecting specific gestures in a semi-public work café environment. Figure 3.1 visualizes the steps involved.

Section 3.1 covers the preprocessing steps applied to the collected data. Section 3.2 focuses on the creation of the dataset. It outlines the process of defining classes, extracting relevant gesture templates, and identifying similar instances within the dataset. This section also describes the data augmentation techniques implemented to enhance dataset diversity and robustness, and it concludes with a summary of the final datasets. Section 3.3 delves into the training pipeline, detailing the application of transfer learning and the hyperparameter tuning process. This section also outlines the experiments conducted to assess the performance of the ST-GCN model.

Figure 3.1: Process Visualization for the Methodology

(Ajar to the Activity Recognition Chain [4])

3.1 Preprocessing

The main objective of the preprocessing step is to improve the quality of the skeleton data sequences, with the goal of increasing comparability for the segmentation phase and potentially improving the performance of the ST-GCN.

The acquired dataset, compressed in 7-Zip format, comprises .txt, .xef, and .pose files. Initially, a script was developed to extract all 5,451 .pose files from these archives. To denoise the data, two filtering methods were applied: a simple moving average (SMA) filter and a Butterworth filter.

The two filters are briefly introduced below:

The simple moving average (SMA) filter works by averaging a subset of adjacent data points within a sliding window. The output is calculated over a window of n points using the following formula:

$$\mathrm{SMA}(t) = \frac{1}{n} \sum_{i=0}^{n-1} x(t-i)$$

For this thesis, a window size of 30 was selected, corresponding to the Kinect camera's sampling frequency of 30 Hz.

A lowpass Butterworth filter is designed to allow signals with frequencies below a certain cutoff frequency to pass while attenuating signals with frequencies above the cutoff. This filter is characterized by its maximally flat frequency response in the passband, ensuring a smooth transition to the stopband without ripples. In this case, a 2^{nd} order Butterworth filter with a cutoff frequency of 2.5 Hz was used.

Figure 3.2 demonstrates the application of these filtering methods on the x-coordinate of the right elbow joint of a skeleton sequence. The figure showcases the raw data along with the filtered outputs. The raw data series exhibits significant jitter, which is effectively reduced by the Butterworth filter, providing a closer approximation to the underlying signal. Although the SMA filter yields the smoothest lines, it also introduces the most alteration to the original data.

The preprocessing step focused on skeleton sequences with durations under 10 minutes to enhance computational efficiency and minimize manual processing, reducing the dataset to 3,827 time series. Both the Simple Moving Average (SMA) and Butterworth filters were applied to these sequences, significantly reducing noise and jitter. This resulted in the creation of three distinct datasets that are now better suited for the subsequent segmentation phase. The refined datasets provide improved comparability between sequences, which is expected to enhance the performance of the Graph Convolutional Network (GCN) during model training and evaluation.

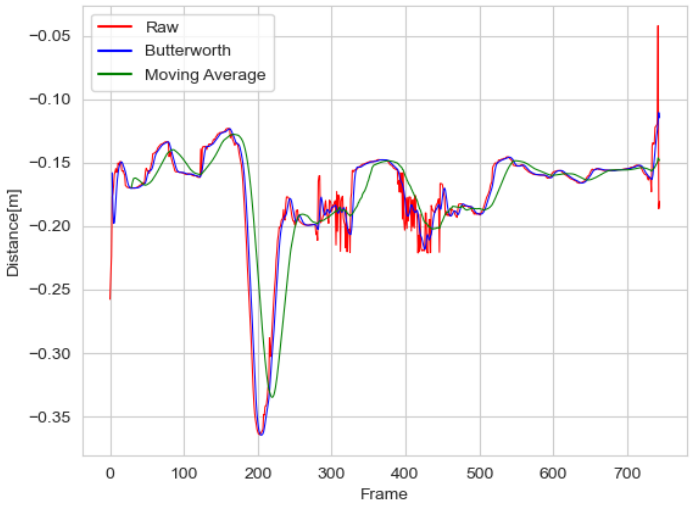


Figure 3.2: Illustration of filter application on the x-coordinate of the elbow joint in a skeleton sequence

3.2 Dataset Creation

As a result of the preprocessing step, three datasets with 3,827 time series have been created. The next step is to label this data and create datasets from it. This process is divided into three main parts as shown in figure 3.3: the labeling, the approach to extract these sequences from the data, and the application of data augmentation techniques to ensure evenly distributed sets.

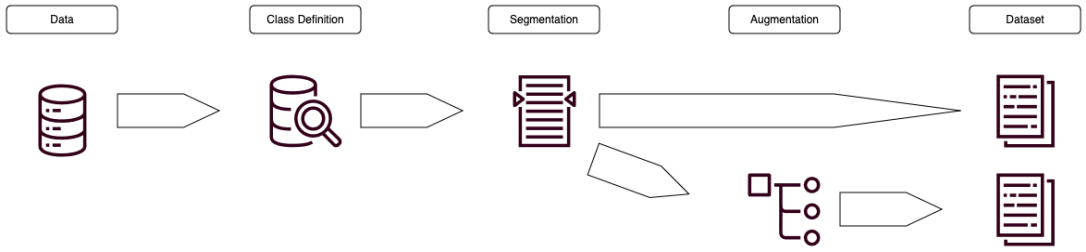


Figure 3.3: Dataset Creation Process

3.2.1 Labeling

As stated in the problem statement, the selection of representative example gestures is essential for the execution of the experiment. A preliminary manual analysis conducted by Jan Schwarzer[1] identified common gestures performed by individuals in the environment. It was observed that many people were either holding or drinking a coffee. Those holding a coffee often moved around the room, while those drinking tended to remain stationary, either sitting or standing.

Consequently, two classes of gestures were defined based on this observed behavior, with each class capable of being performed with either the left or right hand:

1. Holding Gesture: This gesture is characterized by a stable 90° angle between the upper and lower arm. The sequence begins with the subject either picking up a coffee or entering the field of view already holding a coffee, as shown in Figure 3.4.
2. Drinking Gesture: This gesture starts from a holding position, involves reducing the angle between the upper and lower arm, followed by a pause, and then increasing the angle, returning the individual to the starting position, as shown in Figure 3.5

With these gesture classes defined, the next step involves creating a labeled dataset by identifying and extracting these specific motions from the data. This process is detailed in the following section.

3.2.2 Segmentation

Before extracting samples that represent the predefined gesture classes, it is important to develop a process that automatically identifies the sequences within each video that require labeling. The objective is to automate this identification process, to save time and effort.

As demonstrated by Schwarzer et al. [29], Dynamic Time Warping (DTW) is an effective technique for identifying similar behavior patterns within skeleton data. In this thesis, DTW is employed to facilitate the identification and extraction of relevant time series, as illustrated in Figure 3.6.

[1] Ph.D. at the HAW Hamburg and part of the HopE project https://csti.haw-hamburg.de/jan-schwarzer/ (Accessed 2024-09-11)

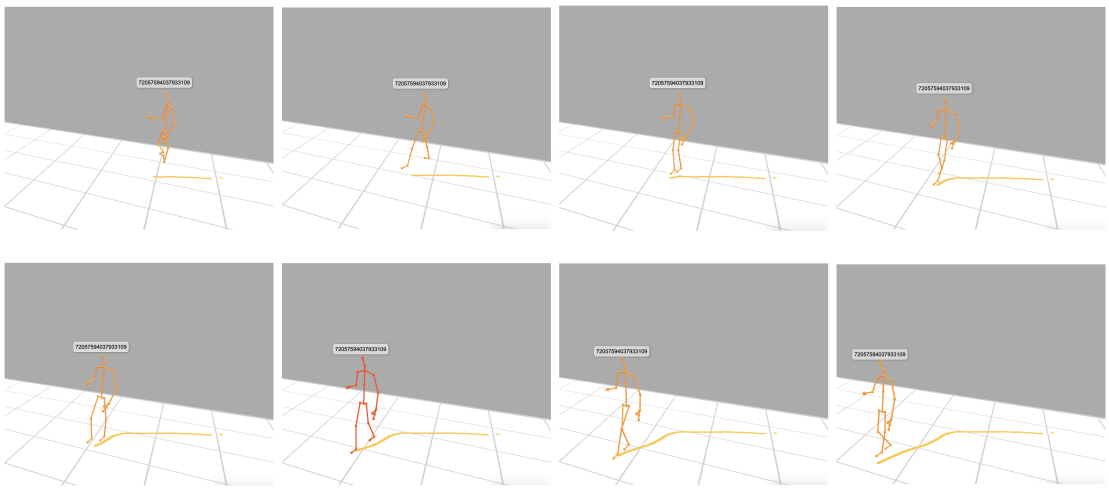


Figure 3.4: Holding gesture

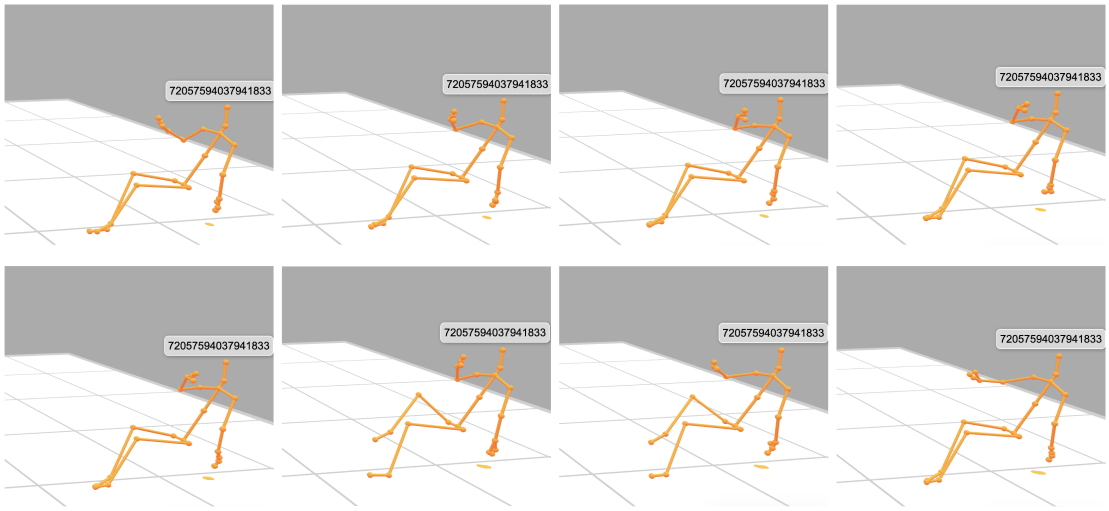


Figure 3.5: Drinking gesture

Through this approach, templates corresponding to the predefined gesture classes are extracted and utilized to find similar sequences within the dataset. The final datasets are then defined based on the evaluation of these extracted sequences.

First, a brief introduction on how Dynamic Time Warping works is provided.

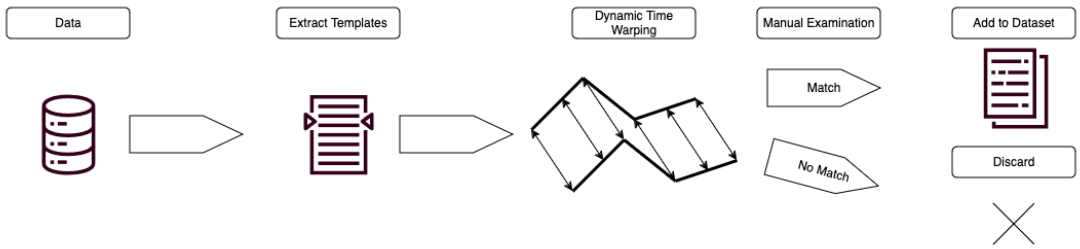


Figure 3.6: Segmentation Process

Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) is a technique used to find the optimal alignment between two time-dependent sequences by applying nonlinear warping. Unlike Euclidean distance, which allows only one-to-one comparisons, DTW enables many-to-one point comparisons, making it more adept at addressing temporal discrepancies. This capability is significant as it allows the identification of shape similarities despite variations in motion speed.

The Algorithm works as follows:

Given two sequences $A := (a_1, a_2, \ldots, a_N)$ of length N and $B := (b_1, b_2, \ldots, b_M)$ of length M, where $a_i = (x_i, y_i)$ and $b_j = (x_j, z_j)$, representing the x and z coordinates of the chosen joint point to make it 2D. The first step is to calculate a distance matrix of size $N \times M$. Each element $d(a_i, b_j)$ is computed using the Euclidean distance:

$$d(a_i, b_j) = \sqrt{\sum_{n=1}^{N} (a_{i,n} - b_{j,n})^2} \tag{3.1}$$

The next step is to calculate a cost matrix, which is used to determine the alignment cost:

$$c_{i,j} = d(a_i, b_j) + \min \begin{cases} c_{i-1,j-1} \\ c_{i-1,j} \\ c_{i,j-1} \end{cases} \tag{3.2}$$

Finally, this cost matrix is used to figure out the warping path w, which is the path with the lowest cost from $c_{M,N}$ to $c_{1,1}$. This is done by starting at $c_{M,N}$ and reversely moving through the cost matrix to find the warping path positions $w = \{c_{N,M}, \ldots, c_{i,j}, \ldots, c_{1,1}\}$. These must be applied to the distance matrix to find the warping path elements $wp = \{d_{N,M}, \ldots, d_{i,j}, \ldots, d_{1,1}\}$. Finally, the DTW distance is calculated by averaging the warping path elements, wp, where k is the number of elements:

$$DTW_d = \frac{1}{k} \sum_{i=1}^{k} wp_i \tag{3.3}$$

Template Selection for Gesture Classes

The following templates were manually selected to represent the defined gesture classes:

1. Walks with a coffee from back right to front left.
2. Sits at the counter and drinks.
3. Stands in front of the screen and drinks.
4. Walks from back to front with a coffee.
5. Sits at the counter, stands up, and walks to the screen with a coffee.

Templates 1, 4, and 5 were categorized as the holding and walking class, while templates 2 and 3 were categorized as the drinking class. These templates were subsequently used as input for the Dynamic Time Warping (DTW) algorithm.

Parameters for Dynamic Time Warping

The DTW algorithm necessitates a reference point to compute the warping path. Given that the gestures were performed with the right hand, the right wrist, and right elbow joint were used as reference points. Each template was processed in three variations: raw, moving average filtered, and Butterworth filtered. Consequently, each template was fed into the algorithm six times, once for each combination of filter and reference point.

An important parameter for the DTW algorithm is the `thresholdDTW`, which ranges from 0 to 1 and defines the maximum warp cost between the template and the time

series. To determine the optimal value for this parameter, different thresholds were tested, starting at 0.075 and incrementing by 0.075 up to 0.3, for both the right elbow and right wrist joints.

Figure 3.7 illustrates the results for the right wrist joint using four distinct threshold values. Both the lower and upper threshold values resulted in either insufficient or excessive matches. The upper threshold values included nearly half of the dataset, depending on the filtering method employed. A similar pattern was observed for the right elbow joint, as shown in Figure A.3. Based on these observations, a threshold value of 0.15 was selected.

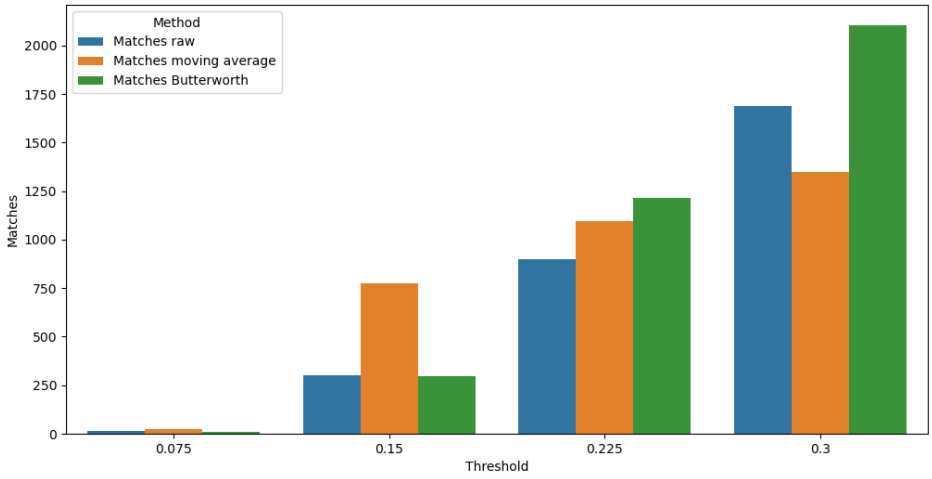


Figure 3.7: DTW result threshold comparison with right wrist as input

Results of Dynamic Time Warping

Figures 3.8 and A.4 illustrate the results of the (DTW) Algorithm across all five gesture templates, using the right elbow and right wrist joints as reference points.

The analysis revealed that the outcomes were largely consistent across the different filtering techniques. Specifically, the templates representing the actions of either standing in front of the screen and drinking or walking from the back right to the front left while holding a coffee produced the highest number of matches.

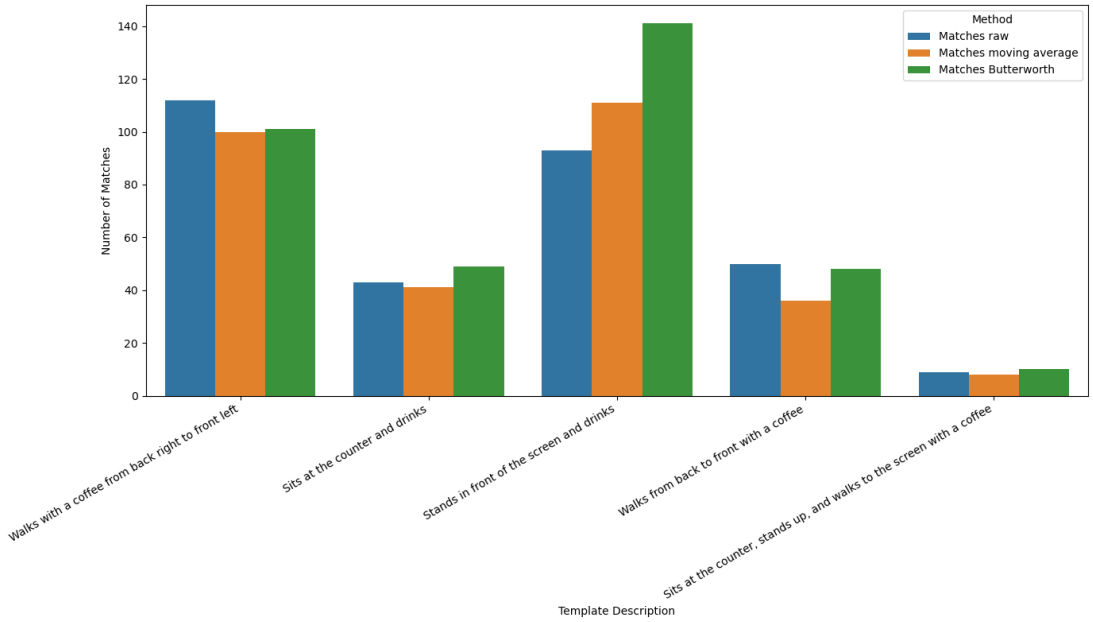


Figure 3.8: DTW results with right elbow as input ($threshholdDTW = 0.15$)

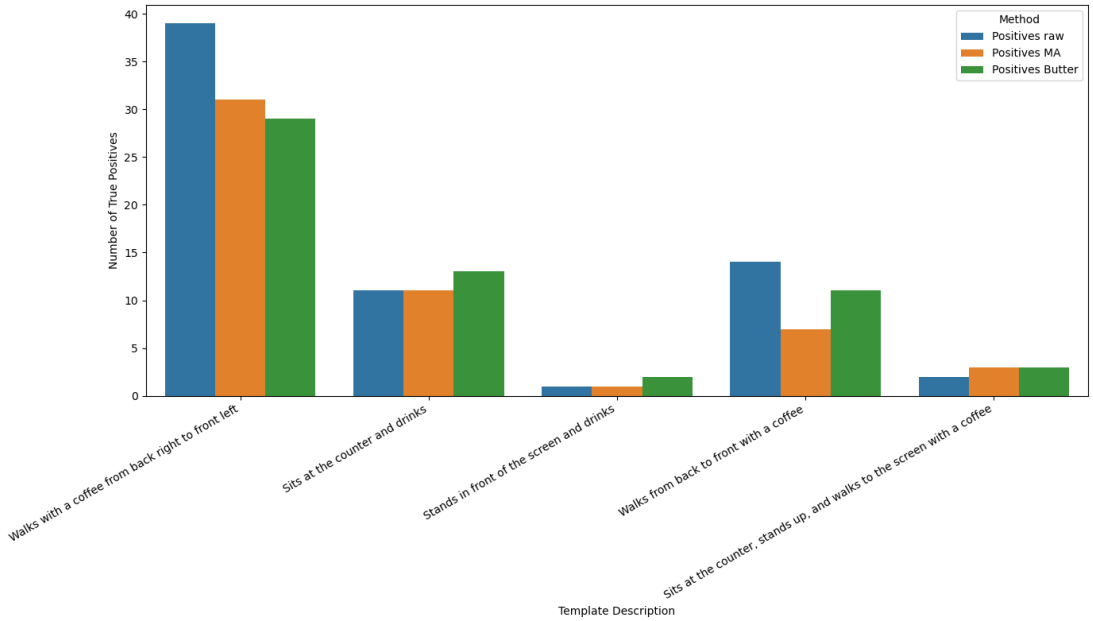


Figure 3.9: True positives of the DTW-Results for the Elbow Joint (`thresholdDTW` = 0.15)

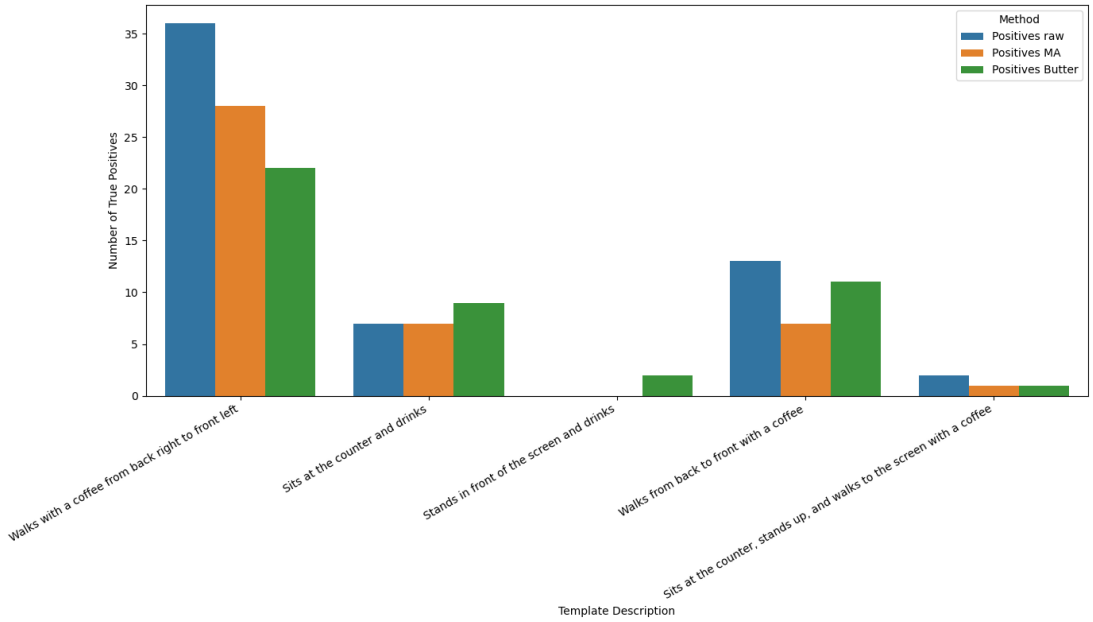


Figure 3.10: True positives of the DTW-Results for the Wrist Joint(`thresholdDTW` = 0.15)

The matches were evaluated manually using the PoseViz tool, and the results are presented in Figures 3.9 and 3.10. These figures display the true positives for each template, highlighting the correctly matched results of the (DTW) algorithm. It was observed that some sequences initially matched to a category had to be reassigned to another.

The evaluation results indicate that the proposed approach performs best on raw data, particularly for the template of a person crossing the room. Overall, the walking templates exhibit the highest true positive rates. However, it is important to note the discrepancy between the number of matches and the true positives.

Figure 3.1 illustrates the overall matches across all templates and joint reference points. As previously described, Templates 1, 4, and 5 are grouped as the holding and walking gesture, while Templates 2 and 3 are grouped as the drinking gesture. A total of 150 samples of individuals crossing and holding a coffee and 54 samples of individuals drinking a coffee were extracted. This extraction was performed manually using a script and the PoseViz tool, which allows for precise timing of the sequences. The desired sequences were segmented and saved using the script.

Class	Matches
Crossing and Holding	150
Drinking	54

Table 3.1: Overall true positives after evaluating the DTW results for the right elbow and wrist joint with `thresholdDTW` = 0.15

Dataset Distribution

To create the non-gesture class, which includes all kinds of movements not classified as the other defined gestures, random sequences were chosen and cut to lengths between 2.5 and 15 seconds. These sequences were then reviewed to ensure they did not represent any of the classified gestures.

To create the non-augmented datasets, a train-validation-test split ratio of 70/15/15 was employed, ensuring that the validation and test data were not augmented. The resulting dataset distributions are as follows:

Class	Train	Validation	Test
Non-Gesture	106	22	22
Crossing and Holding	106	22	22
Drinking	40	7	7

Table 3.2: Distribution of dataset across train, validation, and test sets

Out of this, two non-augmented sets were created for experiments. One set contains only the non-gesture and holding classes, as they are equally distributed, while the other set includes all three classes.

- **Dataset 2 (Non-Augmented, Two-Class)**: This dataset consists of data without augmentation for the two-class problem, excluding the drinking gesture, as distributed in Table 3.2.
- **Dataset 3 (Non-Augmented, Three-Class)**: This dataset consists of raw data without augmentation for the three-class problem, as distributed in Table 3.2.

Each dataset was created three times using the different filtering methods. To now create the finished sets and to distribute the classes, better Data Augmentation steps are employed.

3.2.3 Data Augmentation

The created datasets, particularly Dataset 3, are unbalanced and contain a limited number of samples. To address this, the data is upsampled to increase the dataset size and ensure a balanced class distribution. Following the approach of Rao et al. [28], four augmentation methods were utilized: random rotation, random shear, Gaussian noise, and Gaussian blur. These techniques are briefly introduced as follows:

Random rotation utilizes Euler's rotation theorem, where each axis (X, Y, Z) undergoes matrix multiplication. A main rotation axis is chosen with an angle from $[0, \frac{\pi}{6}]$, while the other two axes rotate between $[0, \frac{\pi}{180}]$. The combined rotation matrix R is obtained by multiplying the individual rotation matrices:

$$R_X(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}$$

$$R_Y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}$$

$$R_Z(\gamma) = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R = R_Z(\gamma)R_Y(\beta)R_X(\alpha)$$

Random shearing is applied using a linear mapping matrix that slants body joints with random angles. The shear transformation matrix is defined as:

$$S = \begin{pmatrix} 1 & s_{YX} & s_{ZX} \\ s_{XY} & 1 & s_{ZY} \\ s_{XZ} & s_{YZ} & 1 \end{pmatrix}$$

where $s_{YX}, s_{ZX}, s_{XY}, s_{ZY}, s_{XZ}, s_{YZ} \in [-1, 1]$ are randomly sampled shear factors.

Gaussian noise is added to simulate noisy positions caused by estimation or annotation errors. The noise is applied to the joint coordinates with a normal distribution $N(0, 0.05)$.

Gaussian blur is applied to smooth noisy joints and decrease action details. The Gaussian kernel value is randomly sampled from $[0.1, 2.0]$ and applied with a sliding window of 15. Joint coordinates are blurred with a 50

$$G(t) = \exp\left(-\frac{t^2}{2\sigma^2}\right), \quad t \in \{-7, -6, \ldots, 6, 7\}$$

To create the augmented datasets, the previously described techniques were applied to improve data distribution for both the two-class and three-class sets.

For the two-class sets, which were initially equally distributed, each sample underwent augmentation, resulting in a fivefold increase in data, thereby producing 530 training samples per class.

For the three-class sets, the non-gesture and crossing and holding classes were first downsampled to 41 samples each. Subsequently, all three classes were augmented by a factor of five, resulting in an equally distributed dataset.

Class	Train (Augmented)	Validation	Test
Non-Gesture	530	22	22
Crossing and Holding	530	22	22

Table 3.3: Distribution of augmented dataset with two classes across train, validation, and test sets

Class	Train (Augmented)	Validation	Test
Non-Gesture	205	22	22
Crossing and Holding	205	22	22
Drinking	205	7	7

Table 3.4: Distribution of augmented dataset with three classes across train, validation, and test sets

The augmented datasets can be defined as follows:

- **Dataset** $A2$ **(Augmented, Two-Class)**: This dataset includes augmented data for the two-class problem set, distributed as shown in Table 3.3.

- **Dataset** $A3$ **(Augmented, Three-Class)**: This dataset includes augmented data for the three-class problem set, distributed as shown in Table 3.4.

3.2.4 Conclusion

The dataset creation process involved several steps to ensure the accuracy and robustness of the data utilized for training and experimental purposes. Initially, representative gesture classes were defined based on observed behaviors, specifically holding and drinking gestures. Samples corresponding to these predefined classes were extracted from the skeleton data using Dynamic Time Warping (DTW), as detailed in Section 3.2.

To mitigate class imbalance and enhance model performance, a range of data augmentation techniques were employed, including random rotation, random shear, Gaussian noise, and Gaussian blur. These augmentation methods increased the dataset size, thereby ensuring a more balanced and representative training set.

Out of this process, twelve datasets were created, named as follows, where r stands for raw, m for moving average, and b for Butterworth:

- **Dataset** $2_r, 2_m, 2_b$ **(Non-Augmented, Two-Class)**: This dataset consists of data without augmentation for the two-class problem, excluding the drinking gesture, as distributed in Table 3.2.
- **Dataset** $3_r, 3_m, 3_b$ **(Non-Augmented, Three-Class)**: This dataset consists of raw data without augmentation for the three-class problem, as distributed in Table 3.2.
- **Dataset** $A2_r, A2_m, A2_b$ **(Augmented, Two-Class)**: This dataset includes augmented data for the two-class problem set, distributed as shown in Table 3.3.
- **Dataset** $A3_r, A3_m, A3_b$ **(Augmented, Three-Class)**: This dataset includes augmented data for the three-class problem set, distributed as shown in Table 3.4.

As a result of this step, a set of datasets has been generated, ensuring a balanced and representative distribution of the gesture classes. These datasets provide the foundation for the next step, where the training of the model will be conducted.

3.3 Training Procedure

This section details the methodology followed to train and evaluate the ST-GCN model. It includes an overview of the model architecture, the use of transfer learning, and a description of the experiments conducted to assess various training approaches and datasets.

3.3.1 Model Architecture

The ST-GCN architecture, illustrated in Figure 3.11, comprises 10 layers of ST-GCN units. The architecture is structured as follows: the first four layers each have 64 output channels, the subsequent three layers each have 128 output channels, and the final three layers each have 256 output channels. The convolutional kernel size is set to 9. To mitigate overfitting, two techniques are employed: the residual network mechanism (also known as skip connections) and a dropout layer. The resulting feature vector is then passed to a Softmax classifier.

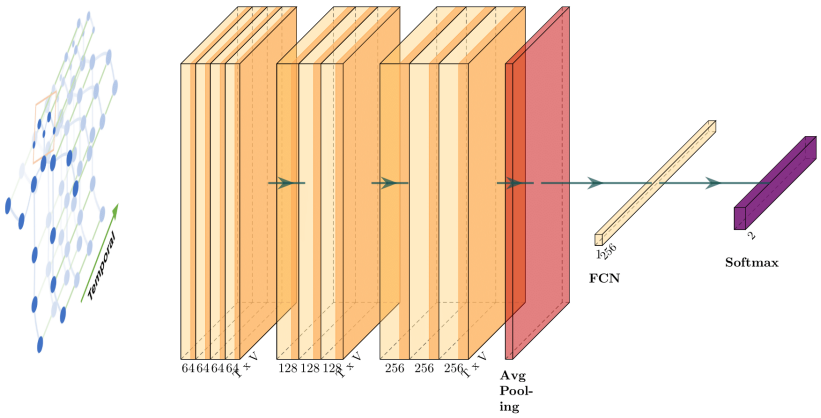


Figure 3.11: ST-GCN Architecture

The ST-GCN model requires input data in the NCTVM format, where:

- N represents the batch size.
- C represents the original node features, which in this case are the triplet coordinates (X, Y, Z).

- T represents the time steps.
- V represents the number of nodes in the graph.
- M represents the number of skeletons in the data record.

For this thesis, the input parameters are defined as follows:

- $V = 25$: The number of nodes in the graph, corresponding to 25 key points in the skeleton data.
- $M = 1$ or 2: The number of skeletons in each data record, which can be 1 or 2 depending on whether transfer learning is applied.
- T: The time steps, corresponding to the time series with the maximum number of frames.

The batch size N will vary depending on the specific experiment parameters. The node features C are derived from the three-dimensional coordinates (X, Y, Z) for each key point in the skeleton data. Each frame in the time series contributes to the time steps T.

The input data for the ST-GCN model can be visualized as a tensor of shape $N \times C \times T \times V \times M$, encapsulating all necessary information for the spatial-temporal graph convolutional operations.

To load the data, each skeleton sequence is transformed from the .pose format to a tensor consisting of (C, T, V, M), where T represents the time series with the highest frame count. All time series with fewer frames are padded with zeros. If transfer learning is applied, which will be introduced in the next section, the number of skeletons M must be 2, as the NTU-RGB+D dataset is trained with classes that include 2 persons. In that case, the frames for the second person are also padded with zeros, as done in the preprocessing for the NTU-RGB+D dataset, ensuring the model is trained on that size.

The tunable parameters of this model include batch size, learning rate, weight decay, dropout rate, and the choice of optimizer, which can be switched between Adam and Stochastic Gradient Descent (SGD). In the original model for the NTU-RGB+D dataset, a dropout rate of 0.5 was chosen, with a weight decay of 0.0001, a base learning rate of 0.1, and a batch size of 64.

The fine-tuning of these parameters will be further elaborated upon after the experiments are introduced.

3.3.2 Transfer Learning

Inspired by the work of Keskes et al. [18], for transfer learning experiments, the ST-GCN model is pretrained using the NTU-RGB+D dataset with the parameters specified in Yan et al. [38]. This pretraining allows the model to learn characteristics that represent general human motion. After the initial training, the first nine layers of the ST-GCN are frozen to retain the learned features. The tenth layer is fine-tuned, with the only modification being the reduction of the output to two or three classes. The network is then fine-tuned using the previously created datasets. This approach aims to leverage the pre-trained knowledge to improve performance and generalization on the specific task at hand.

3.3.3 Experiments

To assess the effectiveness of various training strategies and datasets, eight experiments were designed, incorporating aspects of transfer learning, data augmentation, and different class configurations. These experiments are summarized in Table 3.5. For easier reference in subsequent chapters, the experiment names follow a specific naming convention: T represents the use of transfer learning, A indicates the application of data augmentation, and the numbers 2 or 3 correspond to whether the dataset contains two or three classes, respectively.

Experiment Name	Transfer Learning	Augmentation	Dataset
E_{TA2}	Yes	Yes	$A2$
E_{T2}	Yes	No	2
E_{TA3}	Yes	Yes	$A3$
E_{T3}	Yes	No	3
E_{A2}	No	Yes	$A2$
E_2	No	No	2
E_{A3}	No	Yes	$A3$
E_3	No	No	3

Table 3.5: Summary of Experiments

Each experiment is conducted three times using different preprocessing methods. The design of these experiments aims to isolate the effects of transfer learning and data augmentation on the performance of the ST-GCN model across various datasets. By systematically varying these parameters, the goal is to identify the most effective training strategies and configurations for detecting the defined gestures in in-the-wild data.

3.3.4 Hyperparameter Tuning

To determine the optimal hyperparameters for the experiments, a systematic grid search was employed. This approach not only helps to achieve the best possible performance but also provides valuable insights into the effects of different hyperparameter configurations on the model's outcomes. Once the grid search was completed, each experiment was trained using the optimized parameters in conjunction with three distinct filtering methods.

Grid search is a technique that systematically explores a predefined subset of the hyperparameter space of a learning algorithm. For each combination of hyperparameters, the model is trained and evaluated, allowing for the identification of the optimal configuration based on performance metrics.

The following hyperparameters were tuned: learning rate, batch size, weight decay, dropout rate, and optimizer. The parameter grid was defined as follows:

```
param_grid = {
    'base_lr': [0.01],
    'batch_size': [16, 32],
    'weight_decay': [0.0001, 0.001, 0.01],
    'dropout_rate': [0.1],
    'optimizer': ['SGD', 'Adam']
}
```

As the goal of this thesis is not to seek marginal improvements in model performance, but rather to assess the viability of the approach and identify configurations that produce meaningful results. Consequently, and due to time constraints, the grid search was focused on key parameters identified in preliminary experiments, which suggested that a learning rate of 0.01 and a dropout rate of 0.1 offered a strong starting point.

Thus, the focus was placed on optimizing the remaining parameters: batch size, weight decay, and optimizer. In line with Keskar et al. [17], smaller batch sizes were prioritized to enhance generalization, particularly for smaller datasets.

3.3.5 Conclusion

The process outlined in Figure 3.1 is now fully described. Initially, the data was preprocessed using Butterworth and Moving Average filters. Following this, two classes were defined for labeling. These classes were extracted from the dataset using Dynamic Time Warping, and the corresponding datasets were created. To ensure balanced datasets, data augmentation techniques were applied. The ST-GCN model was then described and used for training. With this, the methodology is complete, and the results are discussed in the following chapter.

4 Evaluation

This chapter presents the results of the experiments conducted in this thesis and provides an evaluation of their results. The experiments were designed to investigate the effectiveness of different hyperparameter configurations, preprocessing techniques, and the impact of transfer learning on gesture recognition performance. By systematically exploring these factors, the goal was to identify the optimal configurations for automating gesture recognition using Graph Convolutional Networks (GCNs).

4.1 Hyperparameter Tuning Results

This section summarizes the results of the hyperparameter tuning process, where the top and mean accuracies, along with key hyperparameters, were evaluated. Table 4.1 presents the top accuracy (Top Acc), mean accuracy (Mean Acc), learning rate (LR), weight decay (WD), dropout rate (DR), batch size (BS), and optimizer for each experiment.

Experiment	Top Acc (%)	Mean Acc (%)	LR	WD	DR	BS	Optimizer
E_{TA2}	93.8	93.2	0.01	0.0001	0.1	16	Adam
E_{T2}	98.2	95.5	0.01	0.0001	0.1	32	Adam
E_{TA3}	91.6	88.7	0.01	0.0001	0.1	16	Adam
E_{T3}	92.4	90.6	0.01	0.001	0.1	16	SGD
E_{A2}	98.1	84.1	0.01	0.01	0.1	32	Adam
E_2	95.0	89.5	0.01	0.001	0.1	32	SGD
E_{A3}	75.4	74.0	0.01	0.01	0.1	16	SGD
E_3	87.5	85.7	0.01	0.0001	0.1	16	SGD

Table 4.1: Results of Hyperparameter Tuning

The optimizer selection showed clear trends across experiments. In the transfer learning experiments, Adam was used consistently in E_{TA2}, E_{T2}, and E_{TA3}, while E_{T3} was the only transfer learning experiment that employed the SGD optimizer. In non-transfer

learning experiments, SGD was the optimizer of choice for E_2, E_{A3}, and E_3, with E_{A2} being the exception, where Adam was applied.

Batch size also varied according to the experimental configuration. For transfer learning, a batch size of 16 was predominantly used in E_{TA2}, E_{TA3}, and E_{T3}, except for E_{T2}, where a batch size of 32 was applied. In non-transfer learning experiments, batch sizes of both 16 and 32 were used. Specifically, two-class problems (E_{A2}, E_2) generally employed a batch size of 32, while three-class problems (E_{A3}, E_3) consistently used a batch size of 16.

The weight decay parameter varied depending on whether transfer learning was used. In transfer learning experiments such as E_{TA2}, E_{T2}, and E_{TA3}, the weight decay was set to 0.0001, except for E_{T3}, where it was 0.001. In non-transfer learning experiments, the weight decay ranged from 0.0001 in E_3 to 0.01 in E_{A2} and E_{A3}, indicating greater variability in these setups.

In summary, the hyperparameter tuning process revealed distinct patterns in optimizer choice, batch size, and weight decay across transfer learning and non-transfer learning experiments. The Adam optimizer was generally used in transfer learning setups, while SGD was more common in non-transfer learning configurations. Batch size remained consistent, with transfer learning often using 16 and non-transfer learning employing either 16 or 32, depending on the classification task. Weight decay was relatively uniform in transfer learning experiments, with more variability observed in non-transfer learning setups.

The overall performance results for these configurations will be discussed in detail in the next section.

4.2 Performance Evaluation Across Experiments

This section presents the mean accuracy outcomes of the defined experiments, which were run with the previously obtained hyperparameters. Each experiment was conducted using three different preprocessing methods: Raw data, Moving Average Filter, and Butterworth Filter.

Early Stopping was applied to halt training when the test accuracy did not improve for 9 consecutive epochs, ensuring efficient model convergence. To ensure consistency and

minimize the impact of randomness, each experiment was repeated five times with each filtering method, and the results were averaged.

The results are displayed in the heatmap shown in Figure 4.1, providing a clear comparison of the mean accuracy across different configurations.

Figure 4.1: Heatmap of the Mean Accuracy across the test datasets

4.2.1 Best and Worst Performing Experiments

The highest accuracy is observed in experiment E_{T2}, where the Butterworth Filter results in an accuracy of 95.9%. The Moving Average Filter and Raw Data both show an accuracy of 95.5%, indicating minimal variation across preprocessing techniques for this experiment.

Conversely, the lowest performance is seen in experiment E_{A3}, where the Moving Average Filter leads to an accuracy of 69.4%. The Butterworth Filter yields a slightly higher accuracy of 74.7%, and Raw Data achieves 74.0%. This demonstrates a notable difference between the Moving Average Filter and the other two techniques.

4.2.2 Impact of Transfer Learning

In experiments involving transfer learning, such as E_{TA2} and E_{T3}, accuracy values tend to be high and consistent across different preprocessing techniques. For example, E_{TA2} achieves 94.1% accuracy with both the Butterworth and Moving Average Filters, with a slight decrease to 93.2% when using Raw Data. Similarly, E_{T3} shows an accuracy of 91.7% with the Butterworth Filter, and 90.6% for both the Moving Average Filter and Raw Data.

In contrast, experiments that do not use transfer learning, such as E_{A2} and E_2, perform worse overall. In E_{A2} the Butterworth Filter results in an accuracy of 83.6%, while the Moving Average Filter reduces this to 82.7%, and Raw Data improves the result to 84.1%. Similarly, in E_2, the Butterworth Filter results in 89.1%, the Moving Average Filter decreases to 86.4%, and Raw Data yields 89.5%.

4.2.3 Impact of Preprocessing Techniques

Across the experiments, the Moving Average Filter generally results in lower accuracy values. This is particularly noticeable in experiments such as E_{A3} and E_3, where the Moving Average Filter produces the lowest accuracy among the preprocessing techniques. For example, in E_3, the accuracy with the Butterworth Filter is 84.5%, while the Moving Average Filter drops it to 76.6%, and Raw Data improves it to 85.7%.

Both the Butterworth Filter and Raw Data exhibit competitive performance across the experiments, each achieving the highest accuracy in four experiments. Specifically, the Butterworth Filter performs best in E_{TA2}, E_{T2}, E_{T3}, and E_{A3}, while Raw Data yields the highest accuracy in E_2, E_3, E_{TA3}, and E_{A2}. Although Raw Data occasionally outperforms filtered data, the differences between the techniques are generally small. The Moving Average Filter, however, consistently delivers the lowest accuracy in six out of eight experiments.

4.2.4 Impact of Data Augmentation

The heatmap reveals that data augmentation consistently results in worse performance across all experiments. In each case where augmentation is applied, such as in E_{TA2},

E_{TA3}, E_{A2}, and E_{A3}, the accuracy is lower compared to experiments without augmentation. For instance, in E_{TA2}, where both transfer learning and augmentation are applied, the accuracy is lower than in E_{T2}, which uses transfer learning without augmentation. Similarly, in E_{A3}, which employs augmentation without transfer learning, the accuracy is significantly lower compared to E_3, which does not use augmentation. This pattern holds across all preprocessing methods.

4.2.5 Effect of Classification Complexity

The results also highlight differences based on the complexity of the classification task, specifically between two-class and three-class experiments. In two-class classification experiments such as E_{T2} and E_2, accuracy is generally higher across all preprocessing techniques. In contrast, three-class experiments such as E_{TA3} and E_{A3} tend to exhibit lower accuracies and show greater sensitivity to the preprocessing method used.

4.2.6 Summary

In summary, the results from the heatmap show that model performance is influenced by several factors, including the use of transfer learning, data augmentation, preprocessing techniques, and the complexity of the classification task. While the Butterworth Filter and Raw Data typically result in higher accuracy, the Moving Average Filter often produces the lowest accuracy. Additionally, experiments involving transfer learning perform more consistently across different preprocessing techniques. Finally, data augmentation consistently leads to lower accuracy in all cases, and the complexity of the classification task—whether two-class or three-class—also plays a role in determining the model's performance.

4.3 Discussion of Results

The primary research question of this thesis was to determine whether Graph Convolutional Networks (GCNs), specifically the ST-GCN model, can be used to automate gesture interpretation in a real-world environment using in-the-wild skeleton data. The results of the experiments show that the ST-GCN model is indeed effective in achieving this goal. This discussion focuses on identifying which configurations, specifically,

the choice of hyperparameters, preprocessing techniques, and transfer learning, can best improve the performance of the ST-GCN model to aid in automating gesture interpretation.

4.3.1 Performance Evaluation

The performance evaluation of the experiments demonstrated that transfer learning consistently led to better results. Experiments E_{TA2}, E_{T2}, E_{TA3}, and E_{T3} achieved higher accuracies across all preprocessing techniques compared to their non-transfer learning counterparts E_{A2}, E_2, E_{A3}, and E_3. This improvement is likely due to the ability of transfer learning to leverage pre-trained features from a similar task, allowing the model to start from a better initialization point and converge faster and more accurately. E_{T2} achieved the highest overall accuracy with 95.9% using the Butterworth Filter, while its non-transfer counterpart E_2 achieved 89.1%. The differences between the transfer learning experiments and their non-transfer learning counterparts range from 6% to 20%, illustrating the clear advantage of transfer learning in improving model performance.

When it comes to preprocessing techniques, the Moving Average Filter consistently resulted in lower accuracies compared to the Butterworth Filter and Raw Data, especially in experiments like E_{A3} and E_3. One possible explanation for this could be that the Moving Average Filter oversmooths the input data, particularly in the case of three-class classification tasks, where the drinking gesture may require more detailed information. This smoothing effect likely diminishes the ability of the model to capture small, precise movements, which are important for identifying subtle gestures. The Butterworth Filter and Raw Data tend to preserve more detailed motion characteristics, which explains their overall better performance across most experiments.

Preprocessing as a whole did not have as significant an impact on performance as initially expected. While there are clear differences between the preprocessing techniques, the margins are relatively small, indicating that the ST-GCN model is quite robust to the choice of preprocessing method. This robustness could be attributed to the model's ability to learn strong spatial and temporal representations of skeletal data, which may reduce its reliance on preprocessing for extracting useful features.

Another finding from the experiments is the consistently negative impact of data augmentation on performance. For instance, E_{TA2}, which applied both transfer learning and augmentation, yielded 1.8% - 2.3% lower accuracy compared to E_{T2}, which did not

use augmentation. Similarly, non-transfer learning experiments like E_{A3} and E_{A2} also experienced a clear decline in performance when augmentation was applied. This drop in accuracy could be attributed to the reality gap introduced by augmenting in-the-wild skeleton data, where synthetic variations may fail to accurately represent the subtleties of real-world gestures[33]. Additionally, the upsampling rate may have been too high, introducing excessive artificial data that confused the model and hindered its ability to generalize effectively.

The comparison between two-class and three-class classification tasks also revealed insights. The two-class tasks, such as E_{T2} and E_2, consistently achieved higher accuracies than their three-class counterparts, like E_{TA3} and E_{A3}. This disparity could be attributed to two factors: class imbalance and the complexity of the gesture classes. In experiments like E_3 and E_{T3}, the presence of an unbalanced dataset could make it harder for the model to distinguish between gestures, leading to lower accuracy. Moreover, the three-class dataset includes the drinking gesture which is more fine-grained, which impose an additional challenge on the model to capture subtle motion differences.

While the results from these experiments are promising, there is still room for improvement. A potential avenue for enhancing performance would be the collection of higher-quality data, possibly through the use of more advanced sensors, such as the ZED X camera. Higher-resolution skeletal data could capture finer motion details and improve the model's ability to generalize across diverse real-world gestures. This is particularly relevant for more intricate gestures, such as the drinking gesture, where the model's performance tends to be lower. This decline in accuracy may partly stem from the limitations in the quality of the current dataset.

4.3.2 Conclusion

In conclusion, the findings demonstrate that the ST-GCN model is effective in detecting specific gestures using in-the-wild skeleton data, making it a valuable tool for automating gesture recognition. Transfer learning has shown to be especially beneficial, substantially improving model performance. Additionally, the choice of optimizer, batch size, and weight decay plays a role in achieving optimal accuracy. Although preprocessing and data augmentation have an impact, their influence is less significant than initially expected. Finally, the complexity of the classification task—whether two-class or three-class—has a clear effect on performance, with simpler tasks consistently yielding better results.

5 Conclusion and Future Work

This thesis investigated whether Graph Convolutional Networks (GCNs), specifically the Spatial-Temporal Graph Convolutional Network (ST-GCN), can aid in the automation of gesture interpretation using in-the-wild skeletal data. While existing research has predominantly focused on datasets collected in controlled laboratory environments, this thesis introduced a comprehensive pipeline for processing and analyzing in-the-wild data. The pipeline involved creating a dataset of diverse gestures from raw, unstructured data, training the ST-GCN model, and evaluating its performance.

The results of the experiments showed that the ST-GCN model achieved good accuracy rates with transfer learning, ranging from 94% to 96%. These results indicate that ST-GCNs, when combined with transfer learning, can effectively interpret gestures in real-world environments. However, it is important to ensure that the dataset is balanced and contains as many samples as possible.

This approach has potential applications beyond the scope of this study, particularly in fields where short-duration input sequences are given. For example, the proposed approach could be used in healthcare monitoring systems to track elderly individuals as they perform daily activities, such as walking up and down stairs, to detect early signs of mobility issues or to alert caregivers in case of dangerous movements. Similarly, this technology could be implemented in rehabilitation programs, where it is important to monitor specific movements and exercises over short durations, such as a patient's ability to lift objects or maintain balance, to assess their recovery progress.

In the industrial sector, this model could be applied to monitor worker safety, particularly during tasks that require precision, such as handling hazardous materials or operating heavy machinery. The short sequence requirement (less than 15 seconds) makes it ideal for detecting quick but significant gestures or behaviors that could indicate safety violations or potential risks. It could also be useful in public spaces for enhancing human-computer interaction, such as in interactive kiosks or smart environments where real-time gesture recognition is essential for an intuitive user experience.

The practical implementation of this model within software tools, such as the PoseViz tool, should also be explored. Integrating the model into such tools would enable real-time recognition of predefined gestures, allowing for more interactive and responsive systems. However, challenges such as managing input sequence length, which should not exceed 400 frames (approximately 15 seconds), must be addressed. Developing an efficient method for identifying and segmenting sequences in real time is essential for successful integration into practical applications.

In conclusion, this thesis has demonstrated the viability of using ST-GCNs for gesture interpretation in real-world settings and has laid the groundwork for future developments in this area. With further refinement of the pipeline and exploration of practical applications across various fields, the approach has the potential to advance the field of gesture recognition in-the-wild.

Bibliography

[1] Ahmad, Tasweer ; Jin, Lianwen ; Zhang, Xin ; Lai, Songxuan ; Tang, Guozhi ; Lin, Luojun: Graph Convolutional Neural Network for Human Action Recognition: A Comprehensive Survey. In: *IEEE Transactions on Artificial Intelligence* 2 (2021), April, Nr. 2, S. 128–145. – URL https://ieeexplore.ieee.org/document/9420299. – Zugriffsdatum: 2024-01-05. – Conference Name: IEEE Transactions on Artificial Intelligence. – ISSN 2691-4581

[2] Alzahrani, Mona S. ; Jarraya, Salma K. ; Salamah, Manar A. ; Ben-Abdallah, Hanêne: FallFree: Multiple Fall Scenario Dataset of Cane Users for Monitoring Applications Using Kinect. In: *2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, URL https://ieeexplore.ieee.org/document/8334766. – Zugriffsdatum: 2024-07-17, Dezember 2017, S. 327–333

[3] Bhatti, Uzair ; Tang, Hao ; Wu, Guilu ; Marjan, Shah ; Hussain, Aamir: Deep Learning with Graph Convolutional Networks: An Overview and Latest Applications in Computational Intelligence. In: *International Journal of Intelligent Systems* 2023 (2023), Februar

[4] Bulling, Andreas ; Blanke, Ulf ; Schiele, Bernt: A tutorial on human activity recognition using body-worn inertial sensors. In: *ACM Computing Surveys* 46 (2014), Januar, Nr. 3, S. 33:1–33:33. – URL https://dl.acm.org/doi/10.1145/2499621. – Zugriffsdatum: 2024-05-24. – ISSN 0360-0300

[5] Chen, Zhenjie ; Wang, Hongsong ; Gui, Jie: Occluded Skeleton-Based Human Action Recognition with Dual Inhibition Training. In: *Proceedings of the 31st ACM International Conference on Multimedia*. New York, NY, USA : Association for Computing Machinery, Oktober 2023 (MM '23), S. 2625–2634. – URL https://doi.org/10.1145/3581783.3612170. – Zugriffsdatum: 2024-01-05. – ISBN 9798400701085

[6] Cheng, Ke ; Zhang, Yifan ; He, Xiangyu ; Chen, Weihan ; Cheng, Jian ; Lu, Hanqing: Skeleton-Based Action Recognition With Shift Graph Convolutional Network. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, URL https://ieeexplore.ieee.org/document/9157077. – Zugriffsdatum: 2024-07-29, Juni 2020, S. 180–189. – ISSN: 2575-7075

[7] Cogollor, José M. ; Hughes, Charmayne ; Ferre, Manuel ; Rojo, Javier ; Hermsdörfer, Joachim ; Wing, Alan ; Campo, Sandra: Handmade task tracking applied to cognitive rehabilitation. In: *Sensors (Basel, Switzerland)* 12 (2012), Oktober, Nr. 10, S. 14214–14231. – ISSN 1424-8220

[8] Cormier, Mickael ; Schmid, Yannik ; Beyerer, Jürgen: Enhancing Skeleton-Based Action Recognition in Real-World Scenarios Through Realistic Data Augmentation, URL https://openaccess.thecvf.com/content/WACV2024W/RWS/html/Cormier_Enhancing_Skeleton-Based_Action_Recognition_in_Real-World_Scenarios_Through_Realistic_Data_WACVW_2024_paper.html. – Zugriffsdatum: 2024-05-27, 2024, S. 290–299

[9] De Souza Santos, Ronnie ; Grillo, William Das N. ; Cabral, Djafran ; De Castro, Catarina ; Albuquerque, Nicole ; França, Cesar: Post-Pandemic Hybrid Work in Software Companies: Findings from an Industrial Case Study. In: *Proceedings of the 2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering*. New York, NY, USA : Association for Computing Machinery, Juni 2024 (CHASE '24), S. 68–78. – URL https://dl.acm.org/doi/10.1145/3641822.3641878. – Zugriffsdatum: 2024-08-08. – ISBN 9798400705335

[10] Eggert, Daniel: Ein System für die Generierung von synthetischen fotorealistischen Bildern mit Prozessen des Deep Learning. . – URL https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/eggert.pdf. – Zugriffsdatum: 2024-08-31

[11] Faity, Germain ; Mottet, Denis ; Froger, Jérôme: Validity and Reliability of Kinect v2 for Quantifying Upper Body Kinematics during Seated Reaching. In: *Sensors (Basel, Switzerland)* 22 (2022), April, Nr. 7, S. 2735. – URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9003545/. – Zugriffsdatum: 2024-07-16. – ISSN 1424-8220

[12] Gupta, Pranay ; Thatipelli, Anirudh ; Aggarwal, Aditya ; Maheshwari, Shubh ; Trivedi, Neel ; Das, Sourav ; Sarvadevabhatla, Ravi K.: *Quo Vadis, Skeleton Action Recognition ?* April 2021. – URL http://arxiv.org/abs/2007.02072. – Zugriffsdatum: 2024-07-29. – arXiv:2007.02072 [cs]

[13] Han, Fei ; Reily, Brian ; Hoff, William ; Zhang, Hao: *Space-Time Representation of People Based on 3D Skeletal Data: A Review.* Februar 2017. – URL http://arxiv.org/abs/1601.01006. – Zugriffsdatum: 2024-08-31. – arXiv:1601.01006 [cs]

[14] Hosna, Asmaul ; Merry, Ethel ; Gyalmo, Jigmey ; Alom, Zulfikar ; Aung, Zeyar ; Azim, Mohammad A.: Transfer learning: a friendly introduction. In: *Journal of Big Data* 9 (2022), Oktober, Nr. 1, S. 102. – URL https://doi.org/10.1186/s40537-022-00652-w. – Zugriffsdatum: 2024-08-25. – ISSN 2196-1115

[15] Johansson, Gunnar: Visual perception of biological motion and a model for its analysis. In: *Perception & Psychophysics* 14 (1973), Juni, Nr. 2, S. 201–211. – URL https://doi.org/10.3758/BF03212378. – Zugriffsdatum: 2024-08-31. – ISSN 1532-5962

[16] Kan, Ruixiang ; Qiu, Hongbing ; Liu, Xin ; Zhang, Peng ; Wang, Yan ; Huang, Mengxiang ; Wang, Mei: Indoor Human Action Recognition Based on Dual Kinect V2 and Improved Ensemble Learning Method. In: *Sensors* 23 (2023), Januar, Nr. 21, S. 8921. – URL https://www.mdpi.com/1424-8220/23/21/8921. – Zugriffsdatum: 2024-07-16. – Number: 21 Publisher: Multidisciplinary Digital Publishing Institute. – ISSN 1424-8220

[17] Keskar, Nitish S. ; Mudigere, Dheevatsa ; Nocedal, Jorge ; Smelyanskiy, Mikhail ; Tang, Ping Tak P.: *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.* Februar 2017. – URL http://arxiv.org/abs/1609.04836. – Zugriffsdatum: 2024-07-29. – arXiv:1609.04836 [cs, math]

[18] Keskes, Oussema ; Noumeir, Rita: Vision-Based Fall Detection Using ST-GCN. In: *IEEE Access* 9 (2021), S. 28224–28236. – URL https://ieeexplore.ieee.org/document/9351913. – Zugriffsdatum: 2024-05-24. – Conference Name: IEEE Access. – ISSN 2169-3536

[19] Kipf, Thomas N. ; Welling, Max: *Semi-Supervised Classification with Graph Convolutional Networks*. Februar 2017. – URL http://arxiv.org/abs/1609.02907. – Zugriffsdatum: 2024-07-16. – arXiv:1609.02907 [cs, stat]

[20] Koch, Michael ; Luck, Kai von ; Schwarzer, Jan ; Draheim, Susanne: The Novelty Effect in Large Display Deployments – Experiences and Lessons-Learned for Evaluating Prototypes, European Society for Socially Embedded Technologies (EUSSET), 2018. – URL https://dl.eusset.eu/handle/20.500.12015/3115. – Zugriffsdatum: 2024-07-19. – ISSN: 2510-2591

[21] Li, Bo ; He, Mingyi ; Cheng, Xuelian ; Chen, Yucheng ; Dai, Yuchao: *Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn*. Juni 2017. – URL http://arxiv.org/abs/1704.05645. – Zugriffsdatum: 2024-01-05. – arXiv:1704.05645 [cs]

[22] Li, Maosen ; Chen, Siheng ; Chen, Xu ; Zhang, Ya ; Wang, Yanfeng ; Tian, Qi: *Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition*. April 2019. – URL http://arxiv.org/abs/1904.12659. – Zugriffsdatum: 2024-07-16. – arXiv:1904.12659 [cs]

[23] Li, Tianjiao ; Liu, Jun ; Zhang, Wei ; Ni, Yun ; Wang, Wenqian ; Li, Zhiheng: *UAV-Human: A Large Benchmark for Human Behavior Understanding with Unmanned Aerial Vehicles*. August 2021. – URL http://arxiv.org/abs/2104.00946. – Zugriffsdatum: 2024-07-17. – arXiv:2104.00946 [cs]

[24] Liu, Ziyu ; Zhang, Hongwen ; Chen, Zhenghao ; Wang, Zhiyong ; Ouyang, Wanli: *Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition*. Mai 2020. – URL http://arxiv.org/abs/2003.14111. – Zugriffsdatum: 2024-01-05. – arXiv:2003.14111 [cs]

[25] Meng, Fanyang ; Liu, Hong ; Liang, Yongsheng ; Tu, Juanhui ; Liu, Mengyuan: Sample Fusion Network: An End-to-End Data Augmentation Network for Skeleton-Based Human Action Recognition. In: *IEEE Transactions on Image Processing* 28 (2019), November, Nr. 11, S. 5281–5295. – URL https://ieeexplore.ieee.org/document/8704987. – Zugriffsdatum: 2024-07-17. – Conference Name: IEEE Transactions on Image Processing. – ISSN 1941-0042

[26] Mobini, A. ; Behzadipour, S. ; Saadat Foumani, M.: Hand acceleration measurement by Kinect for rehabilitation applications. In: *Scientia Iranica* 24 (2017),

Februar, Nr. 1, S. 191–201. – URL http://scientiairanica.sharif.edu/article_4025.html. – Zugriffsdatum: 2024-07-16. – ISSN 2345-3605

[27] Park, Jinyoon ; Kim, Chulwoong ; Kim, Seung-Chan: Enhancing Robustness of Viewpoint Changes in 3D Skeleton-Based Human Action Recognition. In: *Mathematics* 11 (2023), Januar, Nr. 15, S. 3280. – URL https://www.mdpi.com/2227-7390/11/15/3280. – Zugriffsdatum: 2024-07-17. – Number: 15 Publisher: Multidisciplinary Digital Publishing Institute. – ISSN 2227-7390

[28] Rao, Haocong ; Xu, Shihao ; Hu, Xiping ; Cheng, Jun ; Hu, Bin: Augmented Skeleton Based Contrastive Action Learning with Momentum LSTM for Unsupervised Action Recognition. In: *Information Sciences* 569 (2021), April

[29] Schwarzer, Jan ; Fietkau, Julian ; Fuchs, Laurenz ; Draheim, Susanne ; von Luck, Kai ; Koch, Michael: Exploring Mobility Behavior Around Ambient Displays Using Clusters of Multi-dimensional Walking Trajectories. In: *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA : Association for Computing Machinery, April 2023 (CHI EA '23), S. 1–6. – URL https://doi.org/10.1145/3544549.3585661. – Zugriffsdatum: 2024-07-15. – ISBN 978-1-4503-9422-2

[30] Shahroudy, Amir ; Liu, Jun ; Ng, Tian-Tsong ; Wang, Gang: *NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis*. April 2016. – URL http://arxiv.org/abs/1604.02808. – Zugriffsdatum: 2024-01-07. – arXiv:1604.02808 [cs]

[31] Shen, Junxiao ; Dudley, John ; Kristensson, Per O.: *The Imaginative Generative Adversarial Network: Automatic Data Augmentation for Dynamic Skeleton-Based Hand Gesture and Human Action Recognition*. August 2023. – URL http://arxiv.org/abs/2105.13061. – Zugriffsdatum: 2024-07-17. – arXiv:2105.13061 [cs]

[32] Shi, Lei ; Zhang, Yifan ; Cheng, Jian ; Lu, Hanqing: *Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition*. Juli 2019. – URL http://arxiv.org/abs/1805.07694. – Zugriffsdatum: 2024-01-05. – arXiv:1805.07694 [cs]

[33] Sun, Haoyuan ; Azizan, Navid ; Srivastava, Akash ; Wang, Hao: *Private Synthetic Data Meets Ensemble Learning*. Oktober 2023. – URL http://arxiv.org/abs/2310.09729. – Zugriffsdatum: 2024-09-08. – arXiv:2310.09729 [cs]

[34] Veličković, Petar ; Cucurull, Guillem ; Casanova, Arantxa ; Romero, Adriana ; Liò, Pietro ; Bengio, Yoshua: *Graph Attention Networks*. Februar 2018. – URL http://arxiv.org/abs/1710.10903. – Zugriffsdatum: 2024-07-16. – arXiv:1710.10903 [cs, stat]

[35] Wang, Hongsong ; Wang, Liang: *Modeling Temporal Dynamics and Spatial Configurations of Actions Using Two-Stream Recurrent Neural Networks*. April 2017. – URL http://arxiv.org/abs/1704.02581. – Zugriffsdatum: 2024-07-17. – arXiv:1704.02581 [cs]

[36] Wang, Wei ; Zhang, Yu-Dong: A short survey on deep learning for skeleton-based action recognition. In: *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*. New York, NY, USA : Association for Computing Machinery, Februar 2022 (UCC '21), S. 1–6. – URL https://doi.org/10.1145/3492323.3495571. – Zugriffsdatum: 2024-01-05. – ISBN 978-1-4503-9163-4

[37] Xin, Chu ; Kim, Seokhwan ; Cho, Yongjoo ; Park, Kyoung S.: Enhancing Human Action Recognition with 3D Skeleton Data: A Comprehensive Study of Deep Learning and Data Augmentation. In: *Electronics* 13 (2024), Januar, Nr. 4, S. 747. – URL https://www.mdpi.com/2079-9292/13/4/747. – Zugriffsdatum: 2024-06-07. – Number: 4 Publisher: Multidisciplinary Digital Publishing Institute. – ISSN 2079-9292

[38] Yan, Sijie ; Xiong, Yuanjun ; Lin, Dahua: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. New Orleans, Louisiana, USA : AAAI Press, Februar 2018 (AAAI'18/IAAI'18/EAAI'18), S. 7444–7452. – ISBN 978-1-57735-800-8

[39] Yang, Lin ; Zhang, Longyu ; Dong, Haiwei ; Alelaiwi, Abdulhameed ; El Saddik, Abdulmotaleb: Evaluating and Improving the Depth Accuracy of Kinect for Windows v2. In: *IEEE Sensors Journal* 15 (2015), August, Nr. 8, S. 4275–4285. – URL https://ieeexplore.ieee.org/document/7067384/. – Zugriffsdatum: 2024-05-24. – ISSN 1530-437X, 1558-1748, 2379-9153

A Appendix

```
# PoseViz Tracking Data - Schema 0.4
################################################################
# title: Kinect Record '2024-02-22T13:22:03.389'
# owner: JSchwarzer
# organization: HAW Hamburg
# context: Körber Pharma Software GmbH (ehemals Werum IT Solutions GmbH)
# date: 2024-02-22
# id: c992ff67-7db6-4750-9e71-e16422b07a7d
################################################################
# --- camera initialization parameters
# coordinate_system: WORLD
# coordinate_units: UNIT.METER
# --- camera information
# model: Kinect v2 (Kinect for XBOX One)
# unique_id: 006260753647
# depth_max_reliable_distance: 4.5
# depth_min_reliable_distance: 0.5
# vertical_field_of_view (deg): 60
# horizontal_field_of_view (deg): 70.6
# diagonal_field_of_view (deg): 89.5
# tilt_angle (deg): -22
# resolution (depth): 512x424
# fps (depth): 30.0
# --- object detection initialization parameters
# body_format: BODY_FORMAT.KINECT_25
################################################################
```

Figure A.1: Header of a .pose file

```
ts 2024-02-22T13:22:03.637
ct 0.0 0.0 0.0
co -0.134841326071689 0.0 0.0 1.0
f 0
p 72057594037935825 0.2176772 0.4319288 2.570053
m 72057594037935825
k 0 0.189224361614969 0.953144125943348 2.57193797598859
kc 0 tracked
ko 0 -0.04181553 0.976438 -0.2006533 0.0675172
k 1 0.205938262073743 0.65417082849257 2.57616937129179
kc 1 tracked
ko 1 -0.04262648 0.976157 -0.2004826 0.07146461
k 2 0.221240528408946 0.358366193414555 2.5649376673254
kc 2 tracked
ko 2 -0.04343492 0.9664966 -0.2411361 0.07649192
k 3 0.219291691821707 0.196579103855115 2.56304985219842
kc 3 tracked
ko 3 0 0 0 0
k 4 0.383719312807508 0.448675742200171 2.54139924060975
kc 4 tracked
ko 4 0.7320176 -0.6402258 0.2081921 0.1044848
k 5 0.455078972237566 0.668041840500279 2.53354311261377
kc 5 inferred
ko 5 0.8999586 -0.2175638 -0.360632 0.1126286
k 6 0.33605529008808 0.781814195601453 2.36704207101288
kc 6 inferred
ko 6 0.5056814 -0.05437042 0.7824516 -0.3593045
k 7 0.303705532871886 0.795149893162546 2.33769555951576
kc 7 inferred
ko 7 0.4714391 -0.00635652 0.7371585 -0.4840476
k 8 0.0472777403013626 0.450712868438425 2.59874230174075
kc 8 tracked
ko 8 0.7132242 0.6613453 -0.08551468 0.2159189
k 9 -0.0442601288775523 0.730385850360079 2.62320008865406
kc 9 tracked
ko 9 0.7515357 0.2771675 0.5916787 0.09104164
k 10 -0.104531091780819 0.945272754807774 2.58921273385567
kc 10 tracked
ko 10 0.7013751 0.1870934 0.6877959 -0.002375657
k 11 -0.112556533941281 0.985817711181391 2.55600862368344
kc 11 tracked
```

(a) First half of a frame in a .pose file

```
ko 11 0.68137 -0.04171003 0.7163353 -0.1444265
k 12 0.264132232821264 0.94098105525135 2.52722456304635
kc 12 tracked
ko 12 -0.6666797 0.6733426 -0.3104072 0.07612783
k 13 0.297013245258166 1.30913129210478 2.51509851730539
kc 13 tracked
ko 13 0.7486271 -0.1600317 -0.6324465 0.1181468
k 14 0.28109855294098 1.71399230179161 2.55936322745822
kc 14 tracked
ko 14 0.7669988 -0.1438578 -0.5849586 0.2210008
k 15 0.255179342958707 1.77905888607425 2.48317302694624
kc 15 tracked
ko 15 0 0 0 0
k 16 0.1094453587972 94 0.940847546450009 2.55058240996646
kc 16 tracked
ko 16 0.6541067 0.7157809 -0.2413206 0.03957769
k 17 0.0727590660391178 1.23876616812834 2.48485348925487
kc 17 tracked
ko 17 0.6091915 0.1174106 0.7841216 0.01593071
k 18 0.123833549002603 1.6764091665173 2.75021992410412
kc 18 tracked
ko 18 0.5711429 0.3174616 0.6749194 0.3427794
k 19 0.104428457044168 1.70728174808089 2.61615993079738
kc 19 tracked
ko 19 0 0 0 0
k 20 0.217677184462848 0.431928829184456 2.57005328593234
kc 20 tracked
ko 20 -0.04375584 0.9714032 -0.2206769 0.07591378
k 21 0.261073465636597 0.829685062048716 2.31567618596012
kc 21 tracked
ko 21 0 0 0 0
k 22 0.271391865990517 0.78028178063966 2.34929005309085
kc 22 tracked
ko 22 0 0 0 0
k 23 -0.12969789879082 1.0255014398563 2.52996402845049
kc 23 tracked
ko 23 0 0 0 0
k 24 -0.141929352622024 0.970442363346457 2.54291842226212
kc 24 tracked
ko 24 0 0 0 0
e no
```

(b) Second half of a frame in a .pose file

Figure A.2: Representation of a full frame in a .pose file, split between the first half (a) and second half (b).

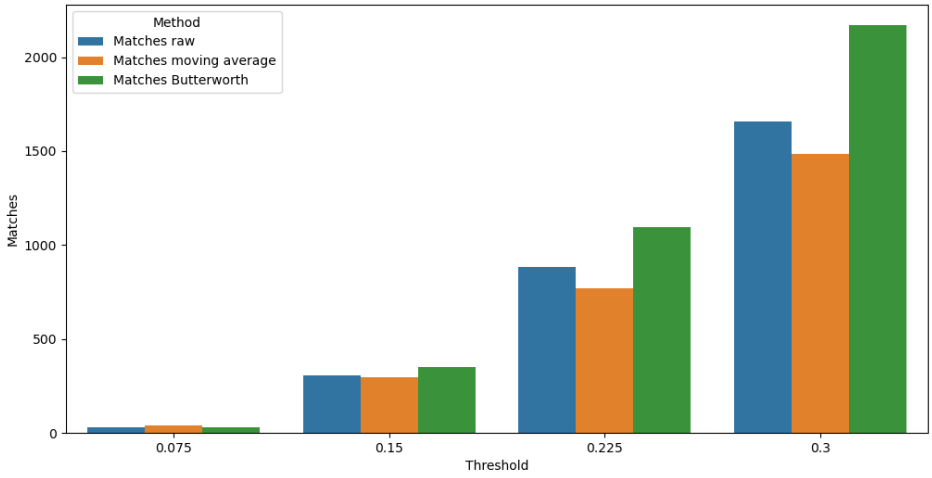


Figure A.3: DTW result threshold comparison with right elbow as input

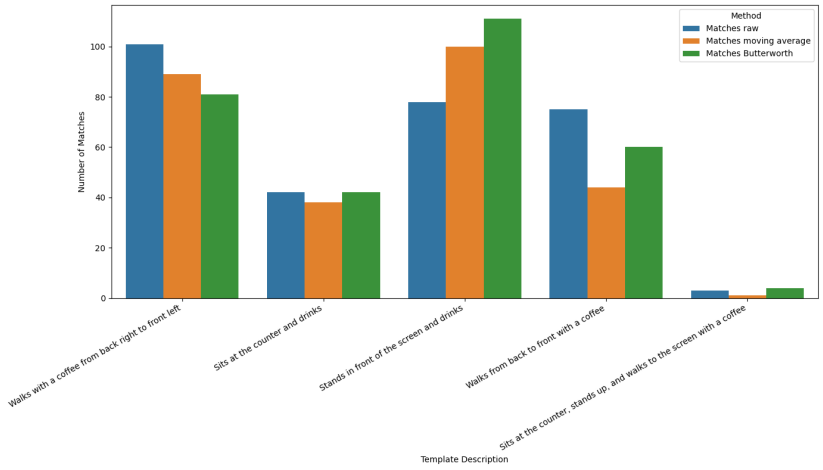


Figure A.4: DTW results with right wrist as input ($threshholdDTW = 0.15$)

Erklärung zur selbständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original