Automatic Classification of Continuous Human Head Gestures

Aman Krishna Satheesh

Master of Science
School of Informatics
University of Edinburgh
2024

Abstract

This study introduces a two-stage pipeline for continuous head gesture recognition using motion capture data, incorporating the HydraMultiROCKET model for both binary gesture detection and multi-class gesture classification, followed by post-processing to convert gesture-wise predictions into gesture segments. The main contributions include the development of the specialized pipeline and incorporation of various methods for post-processing for frame-wise to gesture wise prediction. The proposed approach achieves an average weighted F1-score of 0.57 in cross-validation experiments. The pipeline with HMM-based post-processing technique yields a Cohen's Kappa of 0.61 and an mAP of 0.66, demonstrating its effectiveness in converting frame-wise predictions to accurate gesture segments.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Aman Krishna Satheesh*)

Acknowledgements

I would like to express my sincere gratitude to my dissertation supervisor, Dr. Hiroshi Shimodaira, for his unwavering support, guidance, and encouragement throughout the course of this project. His expertise, insights, and constructive feedback have been invaluable in shaping my research and helping me overcome challenges along the way.

I am also grateful to the previous students whose works have laid the foundation for my research. Their contributions and findings have provided me with valuable insights and inspiration, enabling me to build upon and further advance their work.

Lastly, I am thankful for the constant support and encouragement from my parents, without whom this masters would not have been possible. Their belief in me, constant encouragement, and unconditional love have been the driving force behind my academic pursuits.

Table of Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Research Questions and Objectives 1

1.3 Contributions 2

1.4 Thesis Structure 3

2 Background and Related Work **4**

2.1 Multivariate Time Series Analysis for Motion Capture Data 4

2.1.1 Data Representation and Preprocessing 4

2.1.2 Data Augmentation Techniques 5

2.2 Machine Learning Models for Time Series 5

2.2.1 Convolutional Neural Networks (CNNs) 5

2.2.2 Long Short-Term Memory (LSTM) Networks 6

2.2.3 Transformer Models 6

2.2.4 Hybrid Models 7

2.3 Two-Stage Approaches 9

2.4 Related Work on Head Gesture Recognition 9

3 Methodology **11**

3.1 Dataset Description and Preprocessing 11

3.1.1 Data Structure and Gesture Annotation Protocol 11

3.1.2 Exploratory Data Analysis 11

3.2 Spatio-Temporal Data Preprocessing and Normalization 13

3.2.1 Time Alignment and Spatial Centralization 14

3.2.2 Implementation of Motion-Dependent Data Augmentation . . 14

3.2.3 Rotation Vector Normalization 15

3.2.4 Dataset Preparation for Model Training and Evaluation 16

3.2.5 Train-Test Split Strategy 17
3.3 Proposed Two-Stage Gesture Recognition Pipeline with Post-processing 18
3.4 Implementation of the Gesture Recognition Pipeline 18
3.4.1 Sliding Window Approach for Continuous Recognition . . . 18
3.4.2 Two-Stage Pipeline Architecture 19
3.5 Evaluation Framework 22
3.5.1 Frame-wise Evaluation Metrics 22
3.5.2 Gesture-Wise Evalution 23
3.5.3 Experimental Validation Methodology 23

4 Results and Discussion **24**
4.1 Impact of Motion-Dependent Data Augmentation on Gesture Recognition Accuracy 24
4.1.1 Objective 24
4.1.2 Results 24
4.1.3 Analysis 26
4.2 Comparative Analysis of Binary Gesture Detection Models 27
4.2.1 Objectives 27
4.2.2 Results 27
4.2.3 Analysis 28
4.3 Performance Evaluation of Multi-class Gesture Classification Models 29
4.3.1 Objectives 29
4.3.2 Results 29
4.3.3 Analysis 30
4.4 Comparative Analysis of Post-processing Methods for Gesture Segmentation 30
4.4.1 Objectives 30
4.4.2 Results 31
4.4.3 Analysis 32
4.5 Performance Comparison of Two-Stage and End-to-End Gesture Recognition Pipelines 32
4.5.1 Objectives 32
4.5.2 Results 33
4.5.3 Analysis 34
4.6 Two-Stage Pipeline Performance Evaluation and Cross-Validation . . 35

4.6.1 Cross-Speaker Validation Design for Gesture Recognition Models 35
4.6.2 Objectives 35
4.6.3 Results 35
4.6.4 Analysis 36
4.7 Critical Discussion of Experimental Outcomes 36

5 Conclusion **38**
5.1 Summary of Key Findings and Contributions 38
5.2 Limitations of the Current Approach 39
5.3 Future Research Directions 39

Bibliography **41**

A First appendix **47**
A.1 Sliding Window Experiment Result 47
A.2 TST architecture 47
A.3 Hyper parameter Tuning results 47

Chapter 1

Introduction

1.1 Motivation

Head gestures are a fundamental component of nonverbal communication, providing crucial cues about a person's emotions, attitudes, and intentions. The automatic recognition of head gestures has wide-ranging applications, from enhancing human-computer interaction and virtual reality experiences but despite the advancements in motion capture technology and machine learning, challenges persist in developing robust and accurate head gesture recognition systems that can handle variations in gesture execution across individuals and distinguish between intentional and unintentional movements. This study aims to address these challenges by proposing a two-stage pipeline that leverages state-of-the-art deep learning architectures and data augmentation techniques to improve the performance and generalizability of head gesture recognition using motion capture data.

1.2 Research Questions and Objectives

Previous work by students has established a strong foundation for the development of automatic head gesture recognition systems using motion capture data (Haag and Shimodaira, 2015). (Yang, 2022) focused on the classification and clustering of human head gestures, employing Long Short-Term Memory (LSTM) networks to effectively manage the temporal nature of the data, achieving notable improvements in classification accuracy through dataset expansion and data normalization techniques. Building on this, (Lyu, 2023) addressed the challenge of data imbalance by implementing data augmentation techniques, demonstrating that methods such as noise injection and

temporal warping could significantly enhance the generalization ability of head gesture recognition models, particularly in handling underrepresented gestures. Furthermore, (Lin, 2021) implemented a two-stage classification process that first detects potential gesture segments and then classifies these segments into specific gesture types, which reduced false positives and improved overall accuracy, especially in complex real-world scenarios but despite these advancements, several limitations remain in the current approaches. First, while data augmentation has shown promise, the testing was done only on one model as the paper focused only on augmentation methods. Also, it was not clear how they used augmentation to target minority classes, which can be focused on so the model gets an idea of those classes but adding variability to those gestures using augmentation. Second, the two-stage process, although effective in reducing false positives, did not make use of advanced models and also does not output proper gesture segments with start and end time of gesture as output due to lack of post-processing.

The main objectives of this study are to develop a complete head gesture recognition system which provides start and end time of gestures present in the input data and to investigate various aspects of the pipeline design and implementation and also implement post-processing techniques that convert frame-wise predictions to gesture-wise segment predictions. Specifically, we aim to address the following research questions:

(i) What are the benefits of a two-stage pipeline, which separates gesture detection and classification tasks, compared to one-stage end-to-end models in terms of performance and robustness & generalizability?

(ii) What are effective post-processing techniques to convert frame-wise predictions to gesture-wise segment predictions?

1.3 Contributions

The main contributions of this study include:

- The development of a two-stage pipeline for head gesture recognition using motion capture data, incorporating the HydraMultiROCKET model for both gesture detection and classification.
- A comprehensive evaluation of the proposed pipeline using various performance metrics and cross-validation techniques, demonstrating its effectiveness and

robustness including post-processing technique to convert frame-wise predictions to gesture-wise segment predictions.

- Insights into the impact of data augmentation, model architecture, and post-processing techniques on head gesture recognition performance.

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

- Chapter 2 provides an overview of the background and related work in machine learning approaches, motion capture data, and head gesture recognition.
- Chapter 3 describes the implementation of the proposed two-stage pipeline, including data preprocessing, data augmentation, model architectures, post-processing techniques and evaluation framework.
- Chapter 4 presents the experimental results and analysis, addressing the research questions and objectives outlined in Section 1.2.
- Chapter 5 summaries the main findings and contributions of the study, discusses its limitations, and proposes future research directions.

Chapter 2

Background and Related Work

2.1 Multivariate Time Series Analysis for Motion Capture Data

Time series data contain sequential order with temporal relationships that tell the evolution (dynamics) of systems ordered by time (Liang et al., 2024). So unlike static datasets, where observations are independent of each other and can be analysed independent of each other, time series data have to be analysed keeping the order of data in mind. Standard time series will data points and when each data point has dimensionality of 1 it is called univarite time series and when it is more than 1 it is called multivariate time series which is what we are dealing with in this project.

2.1.1 Data Representation and Preprocessing

Our motion capture data for head gestures consists of multiple time-aligned series, each representing a different aspect of head movement. We call this spatio-temporal data (Yang et al., 2016) as it encompasses two components, spatial component which are three-dimensional coordinates (x, y, z) of the markers including rotation and translation, and second component being the time series component corresponds to the sequence of these spatial positions over time.

for example:

$$X_t = [R_x(t), R_y(t), R_z(t)] \tag{2.1}$$

Where $R_x(t)$, $R_y(t)$, and $R_z(t)$ represent rotations around the x, y, and z axes respectively at time t. We also have translation data in addition to these rotational data. The data we are using is explained in detail in Section 3.1.

2.1.2 Data Augmentation Techniques

Data augmentation is an important method widely used in machine learning, particularly for addressing challenges associated with limited or imbalanced datasets (Khan et al., 2024) which we have faced doing this project. Applying data augmentation for time series data is challenging as each data point is dependent on previous one and changing or altering the data too much can lead to loss of nature of the original data. This becomes more challenging when dealing with mutlivarite time series data as here each dimension is related to each other so creating new data means we need to take into account multiple factors unlike in computer vision problems where simple geometric transformations random cropping and scaling be directly applied to data points (Iwana and Uchida, 2021).

There are multiple methods developed for multivariate time series data augmentation and our particular problem has been explored in this paper (Romain Ilbert, 2024). They focused on impact of augmentation on 13 imbalanced multivariate time series datasets and evaluated the impact on both traditional and deep learning models state-of-the-art time series models. They tried simple methods like noise injection and also complex methods like generating data suing TimeGANS (Time Generative Adversarial Networks) and found simple techniques showed more superior performance compared to complex methods.

2.2 Machine Learning Models for Time Series

2.2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks, originally developed for image processing, have been successfully adapted for time series analysis(Wang et al., 2019). CNNs excel at capturing local patterns and hierarchical features in time series data. A notable CNN-based architecture for time series classification is InceptionTime (Fawaz et al., 2019).

InceptionTime is an ensemble of deep CNN models that incorporates Inception modules, which process input data through parallel convolutional operations with different kernel sizes, capturing patterns at various temporal scales. This multi-scale feature extraction enhances the model's ability to identify relevant patterns in the time series data.

The model also employs residual connections to mitigate the vanishing gradient problem, allowing for more effective training of deeper models (He et al., 2015).

These connections enable the model to learn residual functions, which can be easier to optimize compared to learning the original, unreferenced mapping. We can modify the classification head of the InceptionTime model for binary or gesture classification by adding a global average pooling layer and a sigmoid activation function.

2.2.2 Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory networks, a type of Recurrent Neural Network (RNN), are designed to capture long-term dependencies in sequential data (Hochreiter and Schmidhuber, 1997) whose gating mechanism allows the network to selectively remember or forget information over long sequences. This addresses the vanishing gradient problem common in standard RNNs.

The LSTM architecture comprises a memory cell and three gates: input, output, and forget, work together to regulate the flow of information through the network enabling LSTMs to maintain relevant information over extended periods, making them particularly effective for time series analysis. LSTMs have shown promising results in time series classification, especially when combined with other architectures like (Karim et al., 2018) proposed an LSTM-FCN (Fully Convolutional Network) architecture that leverages both the LSTM's ability to capture temporal dependencies and the FCN's capability to extract local features. LSTMs' effectiveness in handling sequential data and maintaining context over long sequences has led to their widespread use in various time series applications beyond just classification, including prediction, anomaly detection, and sequence generation.

2.2.3 Transformer Models

Transformer models have shown promising results in natural language processing and they have been adapted for time series analysis. The Time Series Transformer (TST) (Zerveas et al., 2020) applies the self-attention mechanism to capture global dependencies in time series data. TST has demonstrated state-of-the-art performance on various multivariate time series classification benchmarks (Zerveas et al., 2020).

The core of the TST architecture is the transformer encoder, which processes input data through multiple self-attention layers. Each layer allows the model to capture complex temporal dependencies and relationships between different variables in the multivariate time series. To maintain the temporal order of the input sequence, TST incorporates positional encodings, which are added to the input embeddings before

passing through the transformer encoder. The final component is a classification head that takes the output of the transformer encoder and produces class probabilities. This typically involves global average pooling followed by a fully connected layer with softmax activation.

The TST model incorporates several key aspects that contribute to its effectiveness in gesture classification like the Multi-head attention allows the model to focus on different aspects of the input sequence simultaneously, capturing complex temporal patterns crucial for gesture recognition (Zerveas et al., 2020). Positional encoding enables the model to understand the temporal order of the input sequence, which is essential for distinguishing between different gestures (Vaswani et al., 2023). Unlike RNNs, transformers can access any part of the input sequence directly, allowing for better modeling of long-range dependencies in gestures (Zerveas et al., 2020).

2.2.4 Hybrid Models

2.2.4.1 HydraMutliRocket

Rocket (Dempster et al., 2020) introduced a novel approach to time series classification using random convolutional kernels where it transforms input time series using a large number of random kernels and applies global max pooling and proportion of positive values (PPV) pooling to the convolution outputs. Rocket achieved state-of-the-art accuracy with significantly lower computational cost than many existing methods as the extra features it generates is then used to train a linear classifier.

MiniRocket (Dempster et al., 2021) further improved on Rocket's efficiency by using a fixed set of kernels and only computing PPV features. MultiRocket (Tan et al., 2022) expanded on MiniRocket by incorporating multiple pooling operators and transformations.

Hydra (Dempster et al., 2022) demonstrated a connection between dictionary methods and convolutional kernel approaches like Rocket.

The HydraMultiROCKET model combines two feature extraction techniques as in this paper (Middlehurst et al., 2024), its shown best performance comes from concatenating features from Hydra with features from MultiROCKET forming HydraMultiROCKET model.

HydraBackbone: The HydraBackbone component processes input data through multiple groups of convolutions with various dilations and paddings. While the MultiRocketBackbone (Tan et al., 2022) component extracts features using random con-

volutional kernels. This backbone processes both the original input and its first-order difference, a technique that has been shown to improve performance in time series classification tasks (Ismail Fawaz et al., 2019). The data flow of the model is shown in Figure 2.1.

The HydraMultiROCKET model incorporates several key aspects that contribute to its effectiveness, including multi-scale feature extraction using dilated convolutions, random kernels for capturing diverse features without the need for training, processing both the input sequence and its first-order difference to capture absolute and relative temporal patterns, and the use of max and min pooling to extract robust features that are invariant to small shifts in the input sequence.

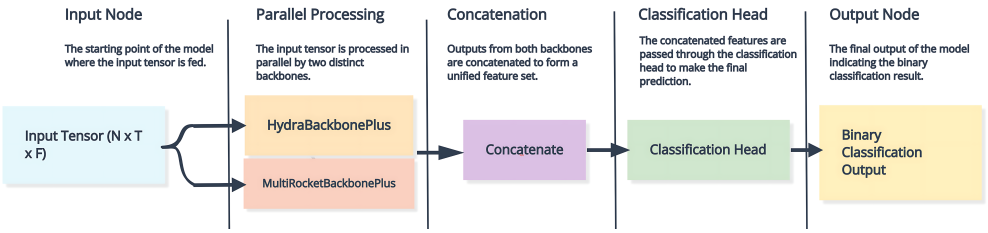

Figure 2.1: Working of HydrMultiRocketPlus

2.2.4.2 ConvTran

Hybrid models aim to combine the strengths of different architectures to better capture the complex nature of time series data. One such model is ConvTran (Foumani et al., 2023), combines convolutional and transformer architectures for multivariate time series classification. The model architecture consists of two main components: the ConvTran-Backbone and the Classification Head. The ConvTranBackbone processes the input data through several layers, starting with an Embedding Layer that uses 2D convolutions to create initial embeddings from mocap data. It then applies both trigonometric absolute position encoding (tAPE) and enhanced relative position encoding (eRPE) to capture absolute and relative temporal information. The Attention Layer incorporates relative position information using a learnable scalar bias matrix. Finally, the Feed-Forward Layer processes the attention output.

The Classification Head includes Global adaptive pooling, flattening, and a linear layer to produce the final classification output. The combination of *tAPE*, *eRPE*, and *Relative Scalar Attention* in the ConvTran model provides significant advantages for head gesture classification using multivariate time series data. This configuration allows

for effective processing of multivariate mocap data, capturing both absolute and relative temporal information crucial for time series classification (Foumani et al., 2023).

2.3 Two-Stage Approaches

Keyword spotting (KWS) systems in ASR (Automatic-Speech Recognition) have evolved from single-stage models to multi-stage architectures to improve accuracy and efficiency. Early deep learning approaches faced challenges balancing detection accuracy and false alarm rates, especially for open-vocabulary tasks. This led to two-stage systems (Zhang et al., 2023) called U2-KWS, using a lightweight first-stage for detection and a more complex second-stage for verification. U2-KWS employs a unified model for initial detection and an attention decoder branch for verification. In this project, we try to replicate this in a basic form for our project.

2.4 Related Work on Head Gesture Recognition

Research on this dataset has been conducted by various students in recent years, and we will discuss some of the significant work that has been done focusing on different aspects of head gesture recognition, including data augmentation, model architecture, and motion capture data analysis.

(Yang, 2022)'s work on the automatic classification and clustering of human head gestures represents a significant contribution to the field. In this study, they utilized Long Short-Term Memory (LSTM) networks to process time-series data from motion capture systems, which captured head gestures. Thier research demonstrated that expanding the dataset and applying normalization techniques could substantially improve the accuracy of gesture classification models. The study achieved a notable improvement in accuracy from 58.78% to 64.19% by increasing the dataset size and the dimensionality of the input data .

Building on this, (Lyu, 2023) explored the use of data augmentation techniques to address the challenge of data imbalance in head gesture recognition tasks. Their implemented basic augmentation methods, such as noise injection and temporal warping, to artificially balance the dataset. The study showed that these techniques could significantly enhance the generalization ability of the models, particularly in handling underrepresented gestures. However, the augmentation techniques applied were relatively simple and did not fully explore the potential of more advanced, motion-dependent

strategies.

(Lin, 2021)'s research introduced a two-stage classification pipeline for head gesture recognition, which first detected potential gesture segments before classifying them. This approach was particularly effective in reducing false positives and improving overall model accuracy, especially in real-world scenarios where gestures are more complex and nuanced. Despite its effectiveness, the study did not employ the most advanced models available, which could potentially limit the system's performance and robustness.

(Wang, 2023)'s study focused on enhancing head gesture recognition by using a hybrid approach that combined Convolutional Neural Networks (CNNs) with LSTM networks. Their work explored the combination of CNNs for feature extraction and LSTMs for sequence modeling, which helped in improving recognition accuracy. The study also delved into the effects of different data augmentation techniques, emphasizing the importance of balancing classes through augmentation.

Another relevant study in this field,(Onuonga, 2023) explored the classification of isolated head gestures using a more sophisticated neural network architecture. This research highlighted the challenges of recognizing subtle and isolated gestures, which are often more difficult to classify due to their lack of context. The study used advanced techniques to preprocess the data and enhance the model's ability to detect and classify these isolated gestures accurately.

Inspired by the prior work on data augmentation and head gesture recognition, the current study implements its own advanced data augmentation techniques, focusing on motion-dependent strategies to improve the balance of gesture classes by focusing on increasing minority class and enhancing model generalization. Additionally, this research addresses the limitations of previous studies that did not utilize state-of-the-art models by incorporating the latest advancements in neural network architectures. By combining these advanced models with a refined classification approach, this study aims to achieve higher f1-score and robustness in head gesture recognition and also develop a post-processing technique to turn frame-wise prediction into gesture-wise segment prediction.

Chapter 3

Methodology

3.1 Dataset Description and Preprocessing

3.1.1 Data Structure and Gesture Annotation Protocol

The dataset, curated by the University of Edinburgh (Haag and Shimodaira, 2015), includes motion capture data from 13 native English speakers (7 male, 5 female) engaged in one-to-one conversations. The speakers exhibit various behaviours (introversion, extroversion, neutral) across multiple recordings. The motion capture system tracks 4 reflective markers on the head, recording 6 degrees of freedom: 3 rotational (R_x, R_y, R_z) and 3 translational (T_x, T_y, T_z) vectors (Bogduk and Mercer, 2000). Data is captured every 10ms, resulting in 6000 data points per minute of recording.

Six main categories of head gestures were annotated shown in Table 3.1. Annotations were manually added by previous students using ELAN software, marking the start and end times of each gesture. The annotations were evaluated for consistency using Cohen's kappa, with an average value of 0.541 indicating good agreement (Cohen, 1960). The raw data and annotations were converted to CSV format for processing, with frame time omitted to prevent overfitting and enhance generalizability across datasets.

3.1.2 Exploratory Data Analysis

Distribution of Gestures: Figure 3.2 presents a stacked bar plot showing the distribution of various gesture types across different actors. The figure shows that actors such as *Adam*, *Beve*, and *Bria* perform the highest number of gestures, each with approximately 400 to 500 gestures recorded. Among these, the *Nodding* gesture emerges as the most frequent across all actors, consistently representing the largest segment within

Type	Description	Most related axis
nd	turning up and down	Z
fu	turning up	Z
fd	turning down	Z
sh	turning left and right	Y
t	turning left or right	Y
ti	tilting left or right	X

Table 3.1: The details of 6 gestures used in this project

each actor's gesture distribution. On the other hand, gestures like *Turning* and *Tilting* appear more evenly distributed among actors, whereas *Face Up* and *Face Down* are less common. The significant variability in the distribution of gesture types among actors suggests that individual differences may play a considerable role in gesture preferences or tendencies, possibly influenced by the context or personal style of each actor.

Duration Of Each Gesture: Figure 3.1 visualizes the distribution of durations (in frames) for various gesture types, such as *Nodding*, *Face Up*, *Face Down*, *Shaking*, *Tilting*, and *Turning*. The *Nodding* gesture shows a moderate duration spread, with an Interquartile Range (IQR) of 50 to 200 frames, but has numerous outliers extending beyond 600 frames. In contrast, *Face Up* has the shortest duration, typically between 50 and 150 frames, with fewer outliers, indicating more consistency. *Face Down* has a wider range, with an IQR from 100 to 300 frames and some outliers exceeding 1000 frames, suggesting occasional extended durations. *Shaking* shows a consistent range within 50 to 200 frames, with fewer outliers, while *Tilting* and *Turning* exhibit moderate spreads, with IQRs from 50 to 200 frames and outliers beyond 500 frames. These patterns indicate that while most gestures fall within typical duration ranges, there are deviations, likely due to contextual or actor-specific factors.

Selecting the optimal duration for input windows is crucial for capturing gestures effectively while minimizing noise and maintaining system responsiveness. The analysis suggests that a window duration between 150 to 200 frames would capture most gestures without missing key movements. A window size of 180 frames can be concluded as good window size, as it balances the central tendency of most gestures, minimizes the influence of outliers, and ensures responsiveness in real-time systems. This duration accounts for the varying lengths of different gestures, providing a balance between completeness and efficiency in gesture recognition. Also looking at the various distribution

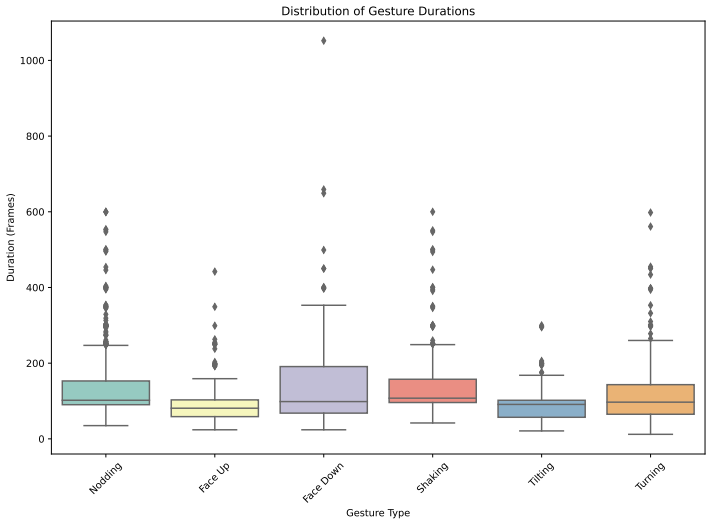

Figure 3.1: Distribution of Gesture Types for Each Actor

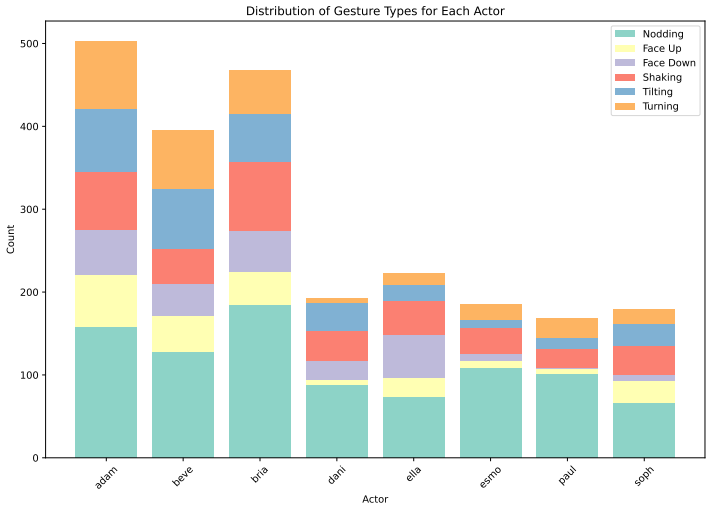

Figure 3.2: Distribution of Gesture Types for Each Actor

of gestures among each actor shows the need for data augmentation which is discussed in the Subsection 3.2.2.

3.2 Spatio-Temporal Data Preprocessing and Normalization

This section explains the data preprocessing that was done. There was no need for data cleaning or filling missing values as seen from Subsection 3.1.2.

The position parameters from the recordings were in .rov file format and the annota-

tions were in .eaf file format. These had to be converted into .csv file using code from previous student (Chen, 2023) so it can be converted to NumPy(Harris et al., 2020) format for efficient data manipulation.

3.2.1 Time Alignment and Spatial Centralization

Two main data preprocessing steps we performed are time alignment and centralization. For time alignment, we map annotations to 6 DoF files using 10ms frames, converting annotation time units from 1ms to 0.1ms and time lag units from 1s to 100ms for consistency.

We apply centralization to each rotated dimension of the signal, removing constant offsets due to varying initial positions or sensor placements, thereby enhancing the model's ability to capture patterns, improve robustness, and focus on relative changes for more accurate analysis and prediction. This was performed using code from the previous student (Chen, 2023).

3.2.2 Implementation of Motion-Dependent Data Augmentation

3.2.2.1 Need for Data Augmentation

Limited annotated recordings and significant class imbalance as seen in Subsection 3.1.2, particularly for nodding (nd) gestures, necessitate data augmentation. This imbalance can lead to biased learning, favouring majority classes (Blagus and Lusa, 2013).

3.2.2.2 Techniques used

This study (Romain Ilbert, 2024) addresses data augmentation for multivariate time series data, as discussed in Section 2.1.2. Following their findings, we employed simple methods for deep learning models, rather than advanced techniques like TimeGAN. We utilized noise injection, rotations, value inversion, scaling, and time interpolation (Núñez et al., 2018). Scaling (randomly between 0.25, 0.5, 1.2, and 1.5 times the original scale) introduces spatial variability. Time interpolation enhances gesture fluidity and temporal granularity while preserving multivariate data relationships and feature interdependencies. To address imbalances in gesture representation (e.g., tilting and turning), we analyzed initial trends in R_y (for turning) and R_x (for tilting) axes. We found that these gestures when done towards one side was classified as a complete gesture thus leading to two different types of same gesture, as illustrated in Figures

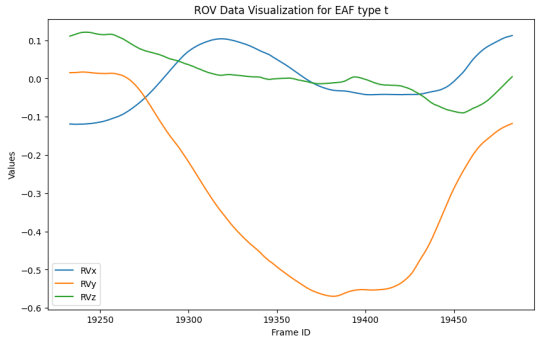

Figure 3.3: Turning Gesture Sample 1

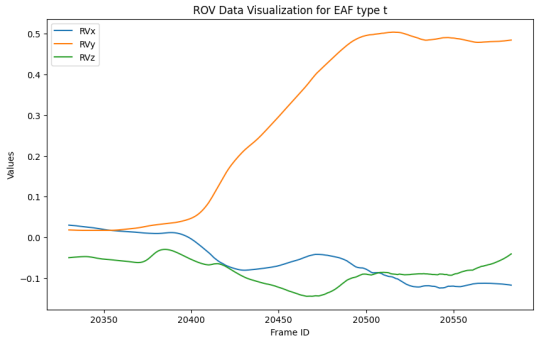

Figure 3.4: Turning Gesture Sample 2

3.3 and 3.4. So we classified segments based on initial increasing or decreasing values and applied augmentation techniques to balance gesture counts in both directions. The augmentation process was implemented per-actor for training data. We extracted individual gestures and randomly applied augmentation techniques according to gesture type showin in Figure 3.7. We retained the non-gesture segments that preceded each selected gesture for augmentation, ensuring the data is consistent to original data but no augmentation was applied to non-gesture segments. Additionally, validation and test datasets remained un-augmented to accurately assess the model's ability to generalise to real-world data.

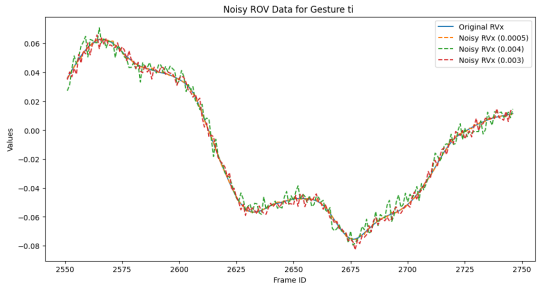

Figure 3.5: Noise Injection Example

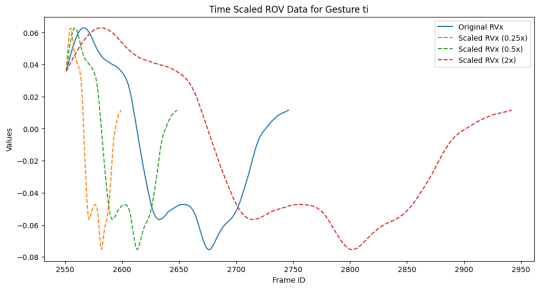

Figure 3.6: Time Scaling Example

3.2.3 Rotation Vector Normalization

We employ the *rotation vector* method (Diebel et al., 2006) to normalize rotational data, offering advantages over Euler angles and quaternions. This method avoids singularities (gimbal lock) associated with Euler angles and doesn't require maintaining a unit norm constraint like quaternions, simplifying computations. The normalization process scales all rotational vectors to unit vectors, focusing on rotation direction while eliminating magnitude variations. This ensures consistency and reliability in subsequent

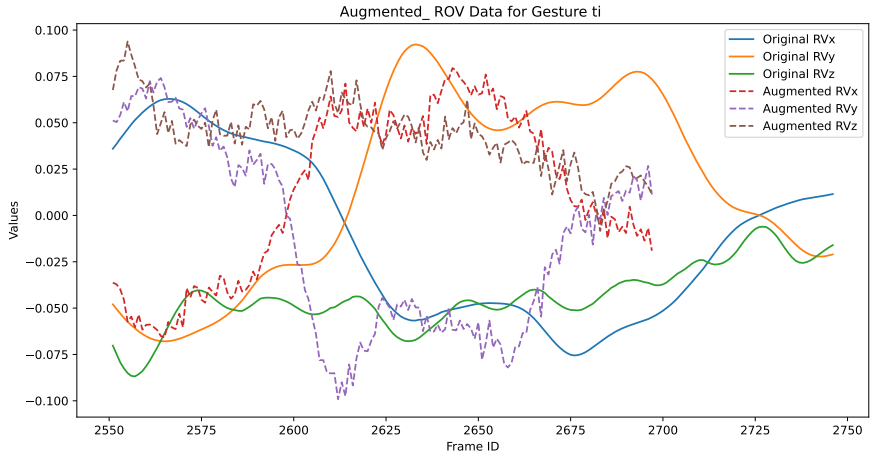

Figure 3.7: Plot showing Original and Augmented gesture for Tilting

data analysis and machine learning models. The magnitude is added an extra feature.

The process of Rotation Vector Normalization is as follows:

1. **Computing Theta**: The magnitude of the rotation vector, θ, is calculated using the Euclidean norm:

$$\theta = \sqrt{RV_x^2 + RV_y^2 + RV_z^2} \tag{3.1}$$

2. **Normalization**: Each component of the rotation vector is then normalized by dividing by θ:

$$RV_{x\_norm} = \frac{RV_x}{\theta}, RV_{y\_norm} = \frac{RV_y}{\theta}, RV_{z\_norm} = \frac{RV_z}{\theta} \tag{3.2}$$

3.2.4 Dataset Preparation for Model Training and Evaluation

After all the required preprocessing steps are performed to get the rotational data in the form and also augmentation applied to training data, now the final dataset needs to be made. This involves matching the rotational data files with the respective annotations and making a combined file. The gesture labels are then one-hot encoded, Table 3.2 shows a sample of the final data format.

After three forms of the data are created.

1. **Dataset-1 for Binary Classification:** This form will only have *ng* gesture label, where 0 means it is a gesture and 1 means it is a non-gesture. This dataset is designed so the binary classification model can train on just identifying if input data has gesture or not. More information about inout data format is talked about in Subsection 3.4.1.

frameid	**R_x_norm**	**R_y_norm**	**R_z_norm**	**theta**	**ng**	**nd**	**fu**	**fd**	**sh**	**ti**	**t**
273	-0.251	-0.134	-0.958	0.058	0	1	0	0	0	0	0
274	-0.253	-0.143	-0.956	0.057	0	1	0	0	0	0	0

Table 3.2

2. **Dataset-2 for Gesture Classification**: This form will have all the gestures labels but will not contain non-gestures. This dataset is purely for the gesture classification model's training so it is able to classify between the 6 gestures. There are no non gestures in this dataset as it is assumed that the input being passed into this model in the two pass pipeline will be data segments that only contain gestures.

3. **Dataset-3 for Pipeline**: This is the base form of the dataset where it contains all the information as seen in Table 3.2.

3.2.5 Train-Test Split Strategy

For splitting the dataset into train, validation and test sets, we are following the speaker-independent style as shown in (Ephrat et al., 2018). This approach ensures that our model learns to generalize across different speakers rather than memorizing specific speaker characteristics. The strategy involves ensuring that data of the same speaker do not appear in more than one set (train, validation, or test). This prevents the model from learning speaker-specific features that could lead to overfitting.

We allocate 4 speakers (57%) to the training set, providing substantial learning data while reserving speakers for validation and testing (Goodfellow et al., 2016). The validation set, comprising 1 speaker (14%), allows for model tuning and hyperparameter optimization on unseen data, preventing overfitting (Bishop, 2006) while the test set, with 2 speakers (29%), offers a robust evaluation of the model's performance and generalization capabilities (Hastie et al., 2009). This split balances the need for sufficient training data with meaningful validation and testing, an approach supported by research on learning from small datasets (Srivastava et al., 2014).

3.3 Proposed Two-Stage Gesture Recognition Pipeline with Post-processing

Our proposed idea for this project combines data augmentation with a two-stage classification pipeline. Initially, to address the challenges of limited and imbalanced datasets we apply data augmentation techniques as suggested by (Romain Ilbert, 2024) to increase the robustness and volume of our training data. Specifically targeting minority class per each actor's recording and creating new data containing more augmented version of the minority gestures.

The core of our methodology is a two-stage classification pipeline inspired by the U2-KWS framework (Zhang et al., 2023). The first stage employs a model for gesture detection, detecting the presence or absence (binary classification) of gestures in time windows. The second stage classifies the specific gesture types in the segments identified by the first stage. Post-processing techniques are applied to refine the outputs into coherent gesture segments. We evaluate different models for two tasks (gesture detection and gesture classification), the pipeline and the post-processing techniques using appropriate metrics. Finally, the best configuration of the pipeline is tested for robustness to different speaker variations using group k-fold validation.

3.4 Implementation of the Gesture Recognition Pipeline

3.4.1 Sliding Window Approach for Continuous Recognition

Sliding windows enhance model efficacy and accuracy (Jaén-Vargas et al., 2022) by capturing local temporal patterns and reducing noise in continuous motion capture data. Based on the analysis in Subsection 3.1.2, a window length of 180 frames was chosen to encapsulate the majority of gestures. Experiments determined that a stride of 10 with this window length provided the optimal balance between accuracy and efficiency (Table A.1). As shown in Figure 3.8, each window is centered on a specific data point, considering both preceding and following data points to better capture subtle temporal patterns and transitions in head gestures.

The output label for each window corresponds to the gesture at the center data point. After the input data is divided into windows, they are all concatenated into 3-D tensor of the shape $X \in R(NTF)$, where N is number of samples (windows), T = 180: temporal dimension (window length) and F = 4: feature dimension.

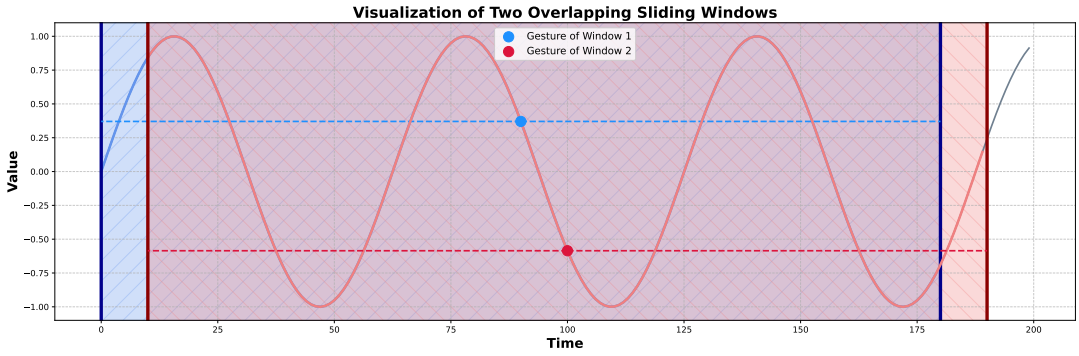

Figure 3.8: Sliding Window Visualisation

3.4.2 Two-Stage Pipeline Architecture

Inspired by the U2-KWS framework for keyword spotting, we propose a novel two-stage , analogous to U2-KWS, pipeline for continuous gesture recognition. The pipeline consists of two main stages: a gesture detection stage (similar to the CTC branch in U2-KWS) and a gesture classification stage (analogous to the attention decoder in U2-KWS), followed by a post-processing step for final output generation.

3.4.2.1 Stage 1: Stage 1: Binary Gesture Detection Model

The first stage is responsible for detecting potential gesture segments within a continuous stream of motion capture data.

Input Processing: The input motion capture recording is divided into overlapping windows of fixed duration. For each window, we store a metadata tuple:

$$M_i = (i, t_{start}, t_{end}, \mathbf{X}_i) \tag{3.3}$$

where i is the window index, t_{start} and t_{end} are the start and end times within the original recording, and $\mathbf{X}_i$ is the raw motion capture data for the window.

Gesture Detection Model: The model outputs a probability p_i for each window, indicating the likelihood of it containing a gesture.

Candidate Selection: Windows with $p_i > \tau_1$, where τ_1 is a predetermined threshold, are marked as potential gesture candidates. We update the metadata for these windows:

$$M_i = (i, t_{start}, t_{end}, \mathbf{X}_i, p_i) \tag{3.4}$$

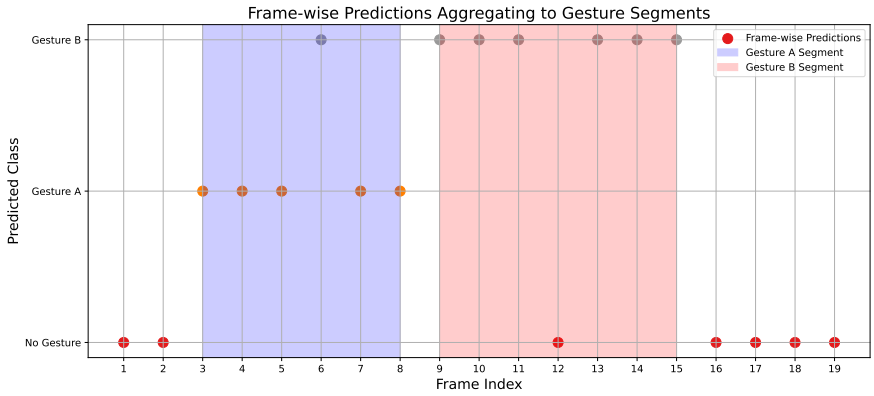

Figure 3.9: Visualisation of Frame-wise predictions to Gesture-Wise Segments

3.4.2.2 Stage 2: Gesture Classification

The second stage, analogous to the attention decoder in U2-KWS, focuses on accurate classification of the gesture candidates identified in the first stage.

Input Refinement: For each gesture candidate, we extract an extended window of data:

$$\mathbf{X}'i = [\mathbf{X}i-k, ..., \mathbf{X}i, ..., \mathbf{X}i+k] \tag{3.5}$$

where k is a context parameter determining the amount of additional data included.

Gesture Classification Model: We utilize the gesture classification model, which, like the attention decoder in U2-KWS, is designed to capture complex spatiotemporal patterns. For each input $\mathbf{X}'_i$, the model outputs a probability distribution $\mathbf{y}_i$ over the set of predefined gesture classes and a ”no gesture” class.

3.4.2.3 Post-processing and Final Output Generation

We require a post-processing step in our pipeline as the model outputs a gesture protection per window it takes an input. We need to post-process this output of frame-wise predictions into gesture-wise segments, visualised in Figure 3.9. In the figure, we can see sometimes the model can output inconsistently in the blue Gesture-A segment with Gesture-B being predicted in between. So it comes down to the post-processing technique to correct this and form correct gesture segments with the added benefit of also giving output as start and end time for each gesture it encounters like in original data annotations.

Our pipeline’s post-processing step is more complex than U2-KWS due to the continuous nature of gesture recognition, taking into account window segments and

finally generating output that needs to be temporally consistent. We experimented with three methods:

1. **Temporal Aggregation and Gesture Segmentation:** Inspired by (Köpüklü et al., 2019), we combine classification results of overlapping windows using a weighted average:
$$\bar{\mathbf{y}}_t = \frac{\sum_i w_i(t)\mathbf{y}_i}{\sum_i w_i(t)} \tag{3.6}$$
where $w_i(t)$ is a weight function based on temporal distance. We identify continuous segments where $\arg\max(\bar{\mathbf{y}}_t)$ remains constant and exceeds a threshold τ_2. For each segment $[t_s, t_e]$, we create a gesture instance: $G_j = (j, t_s, t_e, c_j, s_j)$, where j is the gesture index, $c_j = \arg\max(\bar{\mathbf{y}}t_s)$ is the gesture class, and $s_j = \max t \in [t_s, t_e]\bar{\mathbf{y}}_t[c_j]$ is the confidence score.

2. **Median Filtering:** We apply median filtering to improve spatial-temporal consistency (Pérez and Borz, 2021). The filter uses a window size of 5 and a minimum duration of 10 frames, smoothing noisy predictions and removing short, spurious gesture segments.

3. **Hidden Markov Model (HMM):** We explore HMM-based post-processing to capture temporal dependencies between gestures (Gong et al., 2017). We implement a 7-state HMM where each state represents a gesture class, including a 'no gesture' state. The HMM is trained on ground truth annotations using the Baum-Welch algorithm (Rabiner, 1989), which learns the state transition probabilities (A), emission probabilities (B), and initial state distribution (π). For new data, the Viterbi algorithm finds the most likely state sequence given the frame-wise predictions.

   This sequence is then post-processed to identify continuous runs of the same state, which form our gesture segments. This approach leverages temporal context to smooth inconsistent predictions and produce gesture segments, with the learned transition probabilities capturing typical gesture durations and sequences, while emission probabilities account for uncertainty in the frame-wise classifier's output.

Final Output: The pipeline produces a list of recognized gestures G_j, each with start time, end time, gesture class, and confidence score. Post-processing techniques influences spatial-temporal consistency and overall accuracy of the final output.

3.5 Evaluation Framework

This framework assesses our head gesture recognition pipeline that combines frame-wise classifications from both models into gesture-wise classifications.

3.5.1 Frame-wise Evaluation Metrics

Frame-wise metrics evaluate the binary and classification model performance for individual frames (windows).

3.5.1.1 Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.7}$$

While intuitive, accuracy may be misleading due to class imbalance in our dataset where non-gesture frames likely outnumber gesture frames. It remains a standard metric in gesture recognition literature (Mitra and Acharya, 2007) and serves as a baseline for comparison.

3.5.1.2 Area Under the Receiver Operating Characteristic Curve (AUC-ROC)

$$\text{AUC-ROC} = \int_0^1 TPR(FPR^{-1}(t))dt \tag{3.8}$$

AUC-ROC is valuable for our binary classification task as it's insensitive to class imbalance. It provides insights into the model's ability to discriminate between gesture and non-gesture frames, crucial for our system (Molchanov et al., 2016).

3.5.1.3 F1-score

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.9}$$

The F1-score balances the need to correctly identify gesture frames (precision) with detecting a high proportion of actual gestures (recall). This balance is critical in our task where both false positives and false negatives significantly impact system performance (Yao and Fu, 2014). Given the class imbalance in gesture recognition, we use the macro F1-score to ensure that the performance across all classes, including the less frequent ones, is equally weighted and fairly evaluated (Johnson and Khoshgoftaar, 2019),

3.5.2 Gesture-Wise Evalution

3.5.2.1 Cohen's Kappa

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{3.10}$$

Cohen's Kappa provides a robust measure of segmentation accuracy, accounting for chance agreement to assess overall performance of gesture segmentation in our system, adapting techniques from related fields like dialogue act modeling (Stolcke et al., 2000).

3.5.2.2 F1-score at Intersection over Union (IoU)

$$\text{IoU} = \frac{|\text{Predicted} \cap \text{GroundTruth}|}{|\text{Predicted} \cup \text{GroundTruth}|} \tag{3.11}$$

IoU serves as a critical measure of temporal accuracy in our system. It directly assesses how well the predicted gesture segments align with the ground truth, providing insights into our system's ability to accurately locate gestures in time (Graves et al., 2006). By computing F1-scores at different IoU thresholds, we can assess how our system performs under varying strictness of temporal alignment requirements (Graves, 2012).

3.5.2.3 Mean Average Precision (mAP)

$$\text{mAP} = \frac{1}{|Q|} \sum_{q \in Q} \text{AP}(q) \tag{3.12}$$

The mAP metric provides a comprehensive single-number summary of our head gesture recognition system's performance, encapsulating both the ability to detect gestures and the accuracy of their temporal localization.

3.5.3 Experimental Validation Methodology

Our main experiments employ group k-fold cross-validation (Pedregosa et al., 2011), evaluating model performance across different speaker combinations to assess generalisation capabilities. Each fold assigns unique speaker sets to train, validation, and test, testing gesture recognition on unseen speakers crucial for real-world applicability. Due to computational constraints, we apply cross-validation only in final two evaluations.

Chapter 4

Results and Discussion

4.1 Impact of Motion-Dependent Data Augmentation on Gesture Recognition Accuracy

4.1.1 Objective

The primary purpose of this experiment is to quantitatively assess the impact of data augmentation on the performance and generalizability of our head gesture recognition model. By comparing model performance on augmented versus non-augmented data, we aim to validate the findings of (Romain Ilbert, 2024) in the context of our specific dataset derived from motion capture recordings.

Expected Outcome: We hypothesize that data augmentation techniques will enhance model performance, particularly for underrepresented gesture classes, and improve overall generalization to unseen data. To test this hypothesis, we employ a single-pass classification approach using the InceptionTime architecture, as used in (Romain Ilbert, 2024) and apply group k fold validation with 3 folds (smaller fold chosen to due computational constraints.

The train-test data split for the experiments that follow have been discussed in Subsection 3.2.5 as here Dataset-3 is used.

4.1.2 Results

We evaluated the performance of the InceptionTime model with dropout (0.7) and weight decay on both non-augmented and augmented datasets, training for 20 epochs. The results are presented in Table 4.2 and 4.1 and confusion matrix of predictions are

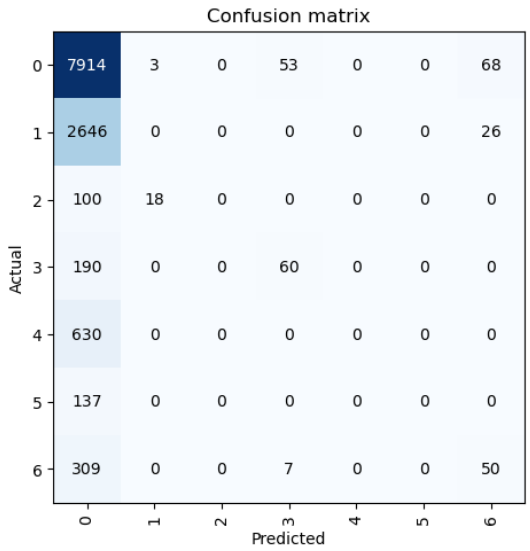

Figure 4.1: Non-Augmnted Dataset Confusion Matrix

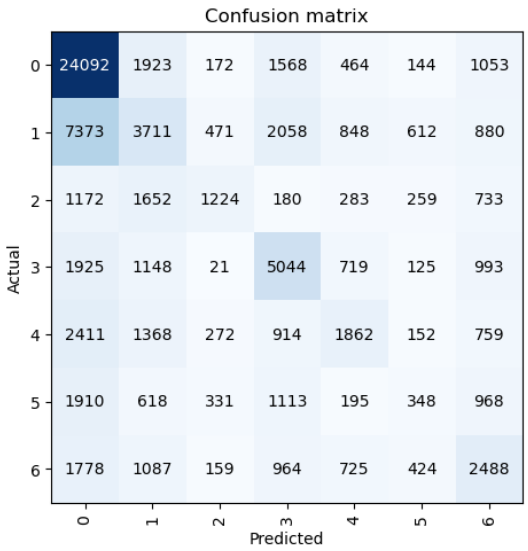

Figure 4.2: Augmnted Dataset Confusion Matrix

visualised in Figures 4.1 and 4.2.

Table 4.1 shows the average precision, recall, and F1-score for the non-augmented dataset after group k fold validation with 3 folds. The 'no gesture' class (0) demonstrates high performance with a precision of 0.66, recall of 0.98, and F1-score of 0.79. However, the model struggles significantly with other gesture types. Classes 1, 2, 4, and 5 (corresponding to nodding, facing up, shaking, and tilting) show zero values across all metrics, indicating complete failure in recognition. Class 3 (facing down) shows moderate performance with a precision of 0.50, but low recall (0.24) resulting in an F1-score of 0.32. Class 6 (turning) also shows limited recognition with an F1-score of 0.20.

In contrast, Table 4.2 presents the metrics for the augmented dataset, revealing substantial improvements across all gesture types. The 'no gesture' class (0) maintains strong performance, albeit slightly reduced, with an F1-score of 0.69. Notably, all previously unrecognized gestures now show measurable performance. Class 1 (nodding) achieves an F1-score of 0.27, class 2 (facing up) reaches 0.30, and class 4 (shaking) attains 0.29. Class 5 (tilting), while still challenging, improves to an F1-score of 0.09. Classes 3 (facing down) and 6 (turning) show significant improvements, with F1-scores increasing to 0.46 and 0.32 respectively.

The confusion matrices in Figures 4.2 and 4.1 provide a visual representation of these improvements. In the non-augmented case (Figure 4.1), we observe a strong bias

Class	**Precision**	**Recall**	F1-**Score**
0	0.66	0.98	0.79
1	0.00	0.00	0.00
2	0.00	0.00	0.00
3	0.50	0.24	0.32
4	0.00	0.00	0.00
5	0.00	0.00	0.00
6	0.35	0.14	0.20

Table 4.1: Non-Augmented Data

Class	**Precision**	**Recall**	**F1-Score**
0	0.59	0.82	0.69
1	0.32	0.23	0.27
2	0.46	0.22	0.30
3	0.43	0.51	0.46
4	0.37	0.24	0.29
5	0.17	0.06	0.09
6	0.32	0.33	0.32

Table 4.2: Augmented Data

Table 4.3: Class : 0 - no gesture (ng), 1 - nodding (nd), 2 - facing up (fu), 3 - facing down (fd), 4 - shaking (sh), 5 - tilting (ti), and 6 - turning (t)

towards the 'no gesture' class, with most predictions concentrated in the first row. The augmented dataset results (Figure 4.2) show a more distributed prediction pattern, with increased correct classifications for all gesture types, as evidenced by higher values along the diagonal.

4.1.3 Analysis

Our experiment demonstrates the positive impact of data augmentation in enhancing InceptionTime model performance for head gesture recognition, validating and extending the findings of (Romain Ilbert, 2024). Data augmentation significantly improved recognition across all gesture classes, addressing the severe class imbalance. In the non-augmented scenario, four out of seven gesture types were unrecognized (F1-score = 0.00). Post-augmentation, all gestures achieved measurable recognition rates (F1-scores: 0.09-0.46).

After augmentation, the 'no gesture' class maintained robust performance with a minor F1-score decrease from 0.79 to 0.69 (-12.7%). Previously unrecognized gestures showed marked improvements: nodding, facing up, and shaking increased to F1-scores of 0.27, 0.30, and 0.29 respectively. Tilting, while still challenging, improved to 0.09. Facing down and turning gestures exhibited substantial gains, with F1-scores increasing by 43.8% (0.32 to 0.46) and 60% (0.20 to 0.32) respectively. The confusion matrices (Figures 4.1 and 4.2) visually confirm the model's enhanced discriminative capability post-augmentation, transitioning from a strong bias towards the 'no gesture' class to a

more balanced distribution of predictions.

These results support our hypothesis that data augmentation enhances model performance, particularly for underrepresented classes, suggesting improved generalization. Augmentation not only allowed the model to be exposed to new variations in the gestures but also helped balance the under-represented classes. However, the variability in improvement across gesture types (e.g., 43.8% increase for facing down vs. minimal improvement for tilting) indicates that certain gestures may require more sophisticated, tailored augmentation techniques.

4.2 Comparative Analysis of Binary Gesture Detection Models

4.2.1 Objectives

The primary objectives of this experiment were to compare the effectiveness of four state-of-the-art time series classification models: TST (Time Series Transformer), InceptionTime, MultiROCKET, and HydraMultiROCKET

Expected Outcome: We hypothesise HydraMultiROCKET to perform better than the rest of the models mainly because inceptionTime and TST are older comparatively and HydraMultiROCKET builds on top of MultiROCKET so HydraMultiROCKET should perform better in theory.

The train-test data split used is Dataset-1 (Subsection 3.2.5).

4.2.2 Results

The performance metrics for each model are summarized in Table 4.4.

Table 4.4 presents the performance metrics for each model. fc_droput is the dropout applied to the fully-connected layer of the model.

We can see base machine learning models like SVM and Logistic Regression model have far lower metrics across the board and an accuracy and f1-score of 56% indicates they are just a bit better than random guessing considering this is a binary classification task. HydraMultiROCKET demonstrated superior performance across all metrics, achieving the highest F1-score (0.844617), accuracy (0.842969), and ROC AUC (0.912257). It outperformed the next best model, MultiROCKET, by margins of 2.4%, 2.5%, and 2.8% in F1-score, accuracy, and ROC AUC respectively.

Model	F1-Score	Accuracy	ROC AUC
SVM	0.564	0.563	0.595
Log-Reg	0.565	0.565	0.599
TST	0.718	0.723	0.797
InceptionTime	0.774	0.777	0.842
MultiROCKET	0.821	0.818	0.884
HydraMultiROCKET	**0.845**	**0.843**	**0.912**

Table 4.4: Performance comparison of binary classification models

InceptionTime showed moderate performance, while TST, despite its sophisticated architecture, lagged behind in all metrics. The estimated F1-score for TST (0.718) aligns with its lower accuracy and ROC AUC scores.

For the HydraMultiROCKET model, we conducted hyperparameter tuning focusing on two key parameters: fc_dropout and weight decay and the results are in Table A.2.

4.2.3 Analysis

HydraMultiROCKET demonstrated superior performance across all metrics, with F1-scores 2.4%–12.7% higher than other models, indicating its suitability for capturing complex temporal patterns in head gesture data. Despite higher computational demands (4GB VRAM vs. 2.5GB for TST), HydraMultiROCKET was chosen for gesture classification due to its accuracy, while a lighter model was used for binary classification. Optimal hyperparameters yielded the highest F1-score (0.844), accuracy (0.842), and ROC AUC (0.912). The effectiveness of higher weight decay (0.1) supports (Zhang et al., 2018)'s research , while the combination of regularization techniques corroborates (Kukačka et al., 2017)'s observations . HydraMultiROCKET's stable performance across configurations (F1-scores: 0.829–0.844) indicates its inherent suitability for binary classification. The optimal configuration with high dropout and weight decay suggests the task benefits from strong regularization due to the subtle nature of head movements.

Model	F1-Score	Accuracy	ROC AUC
MultiRocket	0.35	0.40	0.68
TST	0.43	0.45	0.73
ConvTran	0.46	0.48	0.74
HydraMultiROCKET	0.49	0.51	0.76

Table 4.5: Performance comparison of gesture classification models

4.3 Performance Evaluation of Multi-class Gesture Classification Models

4.3.1 Objectives

The objectives of this experiment were to evaluate the performance of various state-of-the-art models for multi-class gesture classification.

Expected Outcome: We hypothesise ConvTran to perform better but close to performance of HydraMultiROCKET as it combines transformer architecture with CNNs so hybrid model leverages the benefits of both the architecture

The train-test data split used is Dataset-2 (Subsection 3.2.5).

4.3.2 Results

We evaluated four different models on the gesture classification task: MultiRocket, TST (Time Series Transformer), HydraMultiROCKET, and ConvTran. The performance metrics for each model are summarized in Table 4.5.

Table 4.5 presents the performance metrics for the models evaluated. The F1-score ranged from 0.350 to 0.490 across the models. HydraMultiROCKET achieved the highest F1-score (0.490), closely followed by ConvTran (0.460). TST showed moderate performance with an F1-score of 0.430, while MultiRocket had the lowest F1-score of 0.350.

Accuracy scores followed a similar trend, ranging from 0.400 to 0.510. HydraMultiROCKET led with the highest accuracy (0.510), followed by ConvTran (0.480). TST showed moderate accuracy (0.450), and MultiRocket had the lowest accuracy (0.400).

The Area Under the Receiver Operating Characteristic curve (ROC AUC) ranged from 0.680 to 0.760. HydraMultiROCKET again achieved the highest ROC AUC (0.760), indicating its superior ability to distinguish between gesture classes. ConvTran

and TST also performed well, with ROC AUC scores of 0.740 and 0.730, respectively, while MultiRocket had the lowest ROC AUC score of 0.680.

The hyperparameter tuning process focused on adjusting the ‘fc_dropout‘ and ‘weight decay‘ parameters to optimize the performance of the HydraMultiROCKET model. The results of this tuning are summarized in Table A.3.

4.3.3 Analysis

HydraMultiROCKET consistently outperformed other models across all metrics, demonstrating the effectiveness of well-designed convolutional architectures for multi-class gesture classification. ConvTran, a hybrid model combining convolutional and transformer elements, closely followed, indicating the potential of capturing both local and global features in gesture data. The poor performance of ConvTran can be attributed to its nature to perform better on larger data so if our dataset was bigger, it might have shown better performance.

The Time Series Transformer (TST) showed moderate performance, surpassing simpler models but not matching HydraMultiROCKET and ConvTran. This suggests transformer-based architectures may require further adaptation for gesture classification tasks. The better performance of HydraMultiROCKET makes it an ideal candidate for this task

4.4 Comparative Analysis of Post-processing Methods for Gesture Segmentation

4.4.1 Objectives

The objectives of this experiment were to Compare and Evaluate the effectiveness of median filtering, Hidden Markov Model (HMM) based segmentation, and our Temporal Aggregation and Gesture Segmentation post-processing method

Expected Outcome: We hypothesise our Temporal Aggregation and Gesture Segmentation (TAGS) method to perform the best as it uses prediction probabilities to create the gesture segments and for median filtering to perform the worst as it works solely based on the values in the given window without any other information to make an informed decision. We expect the HMM model to have close performance to the TAGS method.

Method	Cohen's Kappa	F1@ IoU=0.1	F1@ IoU=0.25	F1@ IoU=0.5	mAP
Median Filtering	0.57	0.63	0.58	0.53	0.60
TAGS	0.59	0.65	0.62	0.56	0.63
HMM	**0.61**	**0.67**	**0.65**	**0.58**	**0.66**

Table 4.6: Performance comparison of post-processing techniques in the pipeline

The train-test data split used is Dataset-3 (Subsection 3.2.5).

4.4.2 Results

The performance of our pipeline with different post-processing techniques is summarized in Table 4.6. We have called 'Temporal Aggregation and Gesture Segmentation' method as 'TAGS' in the table. The HMM was first trained on some annotations files which were treated as ground truth. These files were of the actors used for training the model and the results in the table 4.6 are from testing on test set.

In our experiments, the TAGS method achieved a Kappa value of 0.59, which decreased slightly to 0.57 with median filtering but increased to 0.61 with the HMM-based approach. These changes in Kappa values suggest that the HMM-based method improves the overall agreement between predictions and ground truth, while median filtering slightly reduces it.

The results show that the F1-scores decrease as the IoU threshold increases, which is expected since higher thresholds pose a greater challenge for accurate gesture localization. However, the relative performance of the post-processing techniques remains consistent across different thresholds. The initial method achieves F1-scores of 0.65, 0.62, and 0.56 at IoU thresholds of 0.1, 0.25, and 0.5, respectively. Median filtering results in lower F1-scores of 0.63, 0.58, and 0.53, while the HMM-based approach yields the highest F1-scores of 0.67, 0.65, and 0.58 at the corresponding thresholds.

These results indicate that the HMM-based method maintains its superiority in terms of gesture recognition performance, even under more stringent evaluation criteria. The TAGS method achieves an mAP of 0.63, which decreases to 0.60 with median filtering but increases to 0.66 with the HMM-based approach.

Median filtering proved to be a simple yet effective technique for smoothing out noisy predictions and improving the spatial consistency of the segmented gestures (Pérez and Borz, 2021)

HMM with 7 states and training it on ground truth annotations, we were able to capture the underlying structure of gesture sequences and make more accurate predictions (Gong et al., 2017)

4.4.3 Analysis

The HMM-based post-processing approach consistently outperformed other methods, achieving the highest Cohen's Kappa (0.61) compared to the TAGS method (0.59) and median filtering (0.57). This indicates better agreement between predicted and ground truth gesture labels. The HMM-based method maintained superior performance across different IoU thresholds, achieving the highest F1-scores compared to the TAGS method and median filtering. This demonstrates better spatial and temporal precision in gesture recognition under stringent evaluation conditions. The TAGS method, incorporating confidence scores, proved more effective than median filtering, emphasizing the importance of prediction confidence in post-processing. The HMM-based technique achieved the highest mAP (0.66), indicating the best balance between precision and recall. These results highlight the significance of appropriate post-processing techniques in gesture recognition. The HMM-based approach, by modeling temporal dependencies and leveraging confidence scores, consistently outperformed other methods across various metrics.

4.5 Performance Comparison of Two-Stage and End-to-End Gesture Recognition Pipelines

4.5.1 Objectives

The main objectives of this experiment were to compare and analyze the performance of the one-stage models with a two-stage pipeline consisting of HydraMultiROCKET models for binary and multiclass classification. The train-test data split used is Dataset-3 (Subsection 3.2.5). We used Group K-Fold (Pedregosa et al., 2011) with with 3 folds and took average for the results discussed in the next subsection.

Expected Outcome: We hypothesise that the two-stage pipeline would outperform the one-stage models and exhibit similar results to the multiclass model since it is the second stage of the pipeline.

Class	0	1	2	3	4	5	6
Pipeline F1-Score	**0.87**	0.42	**0.50**	**0.50**	**0.61**	**0.39**	**0.56**
Model-1 F1-Score	0.86	0.40	0.49	0.46	0.57	0.35	0.51
Model-2 F1-Score	0.83	**0.44**	0.47	0.47	0.59	0.37	0.52

Table 4.7: F1-scores for each gesture class achieved by the two-stage pipeline and one-stage models, where Model-1 is HydraMultiROCKET and Model-2 is ConvTran

Class	0	1	2	3	4	5	6
Pipeline Accuracy	**0.80**	0.43	**0.59**	**0.47**	**0.75**	**0.55**	**0.63**
Model-1 Accuracy	0.76	0.41	0.55	0.46	0.71	0.53	0.59
Model-2 Accuracy	0.78	**0.44**	0.56	0.43	0.73	0.50	0.61

Table 4.8: Accuracy scores for each gesture class achieved by the two-stage pipeline and one-stage models, where Model-1 is HydraMultiROCKET and Model-2 is ConvTran

4.5.2 Results

Table 4.7 presents the F1-scores for each gesture class achieved by the two-stage pipeline, the baseline one-stage model, and a third one-stage model. The two-stage pipeline outperforms the one-stage models in most gesture classes, with notable improvements in classes 2, 3, 4, 5, and 6. The third one-stage model achieves the highest F1-score for class 1, while the baseline model performs slightly better in class 0.

Table 4.8 shows the accuracy scores for each gesture class. The two-stage pipeline consistently achieves higher accuracy compared to the one-stage models across all classes. The third one-stage model exhibits better accuracy than the baseline in classes 1 and 4, while the baseline model performs slightly better in class 3.

Table 4.9 summarizes the overall performance metrics, including Cohen's Kappa, F1-scores at different IoU thresholds, and mAP. The two-stage pipeline achieves the highest Cohen's Kappa (0.61), indicating better agreement between the predicted and ground truth gesture labels. The pipeline also outperforms the one-stage models in

Metrics	Cohen's Kappa	F1@ IoU=0.1	F1@ IoU=0.25	F1@ IoU=0.5	mAP
Pipeline	0.61	0.67	0.65	0.58	0.66
Model-1	0.58	0.65	0.62	0.56	0.65
Model-2	0.57	0.64	0.60	0.55	0.65

Table 4.9: Overall performance metrics for the two-stage pipeline and one-stage models

terms of F1-scores at all IoU thresholds, demonstrating its superior spatial and temporal precision in gesture recognition. The mAP scores are comparable across all models, with the two-stage pipeline achieving a slightly higher value (0.66) compared to the one-stage models (0.65).

4.5.3 Analysis

The experimental results support our hypothesis that the two-stage pipeline outperforms the one-stage models in gesture recognition. The pipeline's superior performance can be attributed to its design, which separates the binary classification task from the multiclass classification task (Köpüklü et al., 2019), allowing the models in each stage to specialize in their respective tasks.

The higher F1-scores, accuracy, Cohen's Kappa, and F1-scores at different IoU thresholds achieved by the two-stage pipeline demonstrate its ability to better distinguish between gestures, reduce false positives and negatives, and improve spatial and temporal localization. Although the mAP scores are comparable across all models, the slightly higher value achieved by the two-stage pipeline indicates its overall better performance in terms of precision and recall.

In conclusion, the two-stage pipeline approach proves to be more effective than the one-stage models for gesture recognition. The pipeline's design, which separates the binary and multiclass classification tasks, allows for specialization and improved performance in each stage, demonstrating its potential for accurate and robust gesture recognition.

Fold	Train F1	Val F1	Test F1
1	0.69	0.58	0.61
2	0.67	0.56	0.57
3	0.68	0.59	0.60
4	0.65	0.55	0.56
5	0.63	0.53	0.54
6	0.62	0.52	0.53
7	0.64	0.54	0.55
Average	0.66	0.56	0.57

Table 4.10: Group k-fold cross-validation results with all possible speaker combinations (weighted F1-score)

4.6 Two-Stage Pipeline Performance Evaluation and Cross-Validation

4.6.1 Cross-Speaker Validation Design for Gesture Recognition Models

To evaluate the pipeline's performance under different speaker configurations, we conducted group k-fold cross-validation (Pedregosa et al., 2011) with k=7, considering all possible speakers come at least once in the test set. In each fold, four speakers were used for training, one speaker for validation, and the remaining two speakers for testing.

4.6.2 Objectives

The objectives of this analysis were to perform group k-fold cross-validation to assess the pipeline's generalization ability across different speaker configurations.

Expected Outcome: We hypothesise the pipeline to show good generalizability towards the different folds due to the augmentation we have performed.

The train-test data split used is Dataset-3 (Subsection 3.2.5).

4.6.3 Results

The results of the group k-fold cross-validation with all possible speaker combinations are presented in Table 4.10.

The group k-fold cross-validation results with all possible speaker combinations show variations in the pipeline's performance across different speaker configurations for training, validation, and testing sets. The average weighted F1-score for the training set is 0.66, indicating a reasonable performance on the speakers used for training. For the validation set, the average weighted F1-score is 0.56, suggesting a drop in performance when evaluating on unseen speakers. The average weighted F1-score for the testing set is 0.57, which is lower than the initial testing configuration (F1-score: 0.65).

The highest test F1-score of 0.61 is achieved in Fold 1, while the lowest test F1-score of 0.53 is observed in Fold 6. These variations in performance highlight the impact of speaker characteristics and gesture variations on the pipeline's generalization ability.

4.6.4 Analysis

The group k-fold cross-validation analysis using weighted F1-scores revealed limitations in the pipeline's generalization ability across different speaker configurations and gesture classes. The lower average test F1-score compared to the initial testing configuration indicates reduced performance when considering a wider range of speaker combinations. The weighted F1-score provided a balanced evaluation across all classes, ensuring minority class performance was not overshadowed. The consistently lower validation set performance (average F1-score: 0.56) compared to the training set suggests potential overfitting, emphasizing the need for effective regularization, data augmentation, and careful model selection. The analysis highlights the need for a diverse and representative dataset, effective regularization techniques, and exploration of advanced methods like transfer learning and data augmentation.

4.7 Critical Discussion of Experimental Outcomes

Our study significantly contributes to head gesture recognition by comprehensively analyzing a two-stage pipeline for continuous gesture recognition. In contrast to (Yang, 2022), which focused on classification and clustering using LSTMs, our study explores data augmentation, compares state-of-the-art architectures, and evaluates post-processing methods. Our augmentation strategies enhance generalization, particularly for underrepresented classes, achieving F1-scores around 0.60, indicating balanced performance compared to the accuracy of 64.19% in (Yang, 2022).

Building upon (Lyu, 2023)'s work on data imbalance, our two-stage pipeline sepa-

rates gesture detection and classification, similar to (Lin, 2021). However, our approach incorporates the latest advancements in neural network architectures, demonstrating better performance with an average F1-score of 57.00% after cross-validation, compared to 47.13% in (Lin, 2021). Our pipeline also outperforms the hybrid approach of (Wang, 2023), achieving an average F1-score of 0.57 compared to their range of 0.52 to 0.71.

Our study introduces a post-processing technique using HMMs to convert frame-wise predictions into gesture-wise segments, enhancing spatial-temporal consistency and accuracy with a Cohen's Kappa of 0.61 and mAP of 0.66. The experimental results address our research questions:

(i) The two-stage pipeline outperforms one-stage models, suggesting that specialized stages lead to better performance, with an average F1-score of 0.57.

(ii) HMM-based post-processing effectively converts frame-wise predictions to gesture-wise segments, achieving a Cohen's Kappa of 0.61 and mAP of 0.66, outperforming other techniques.

The proposed approach demonstrates strengths in its two-stage design, advanced architectures, data augmentation, and post-processing techniques. However, it faces challenges in generalization across speakers and classes, sensitivity to gesture variations, distinguishing between intentional and unintentional (non-gesture) movement. Future research should focus on diverse datasets, advanced machine learning techniques, and optimization strategies to enhance robustness and real-world applicability discussed further in Section 5.3.

Chapter 5

Conclusion

5.1 Summary of Key Findings and Contributions

This study introduces a novel two-stage pipeline for head gesture recognition using motion capture data, consisting of a HydraMultiROCKET model for binary gesture detection and multi-class classification, followed by post-processing. The main contributions include the development of a specialized pipeline, the investigation of data augmentation techniques, the incorporation of HMMs as a post-processing technique, and a comprehensive evaluation using various performance metrics and cross-validation techniques. The experiments conducted address the two main research questions:

(i) The first question, regarding the benefits of a two-stage pipeline compared to one-stage end-to-end models, is answered through the comparison of the proposed two-stage approach with one-stage models. The results demonstrate that the two-stage pipeline, which separates gesture detection and classification tasks, outperforms one-stage models in terms of performance, robustness, and generalizability. This improvement is attributed to the pipeline's ability to optimize each task independently, allowing for the use of specialized models for binary gesture detection and another model for multi-class classification.

(ii) The second question, concerning effective post-processing techniques for converting frame-wise predictions to gesture-wise segment predictions, is addressed through the incorporation of HMMs in the pipeline. The HMM-based approach proves to be a powerful tool for capturing temporal dependencies between gestures and generating gesture segments with start and end times. This post-processing technique significantly enhances the spatial-temporal consistency and overall

accuracy of the system, addressing a key limitation in previous studies that did not focus on producing gesture segments.

Furthermore, the study investigates advanced, motion-dependent data augmentation techniques, such as noise injection, rotations, and time interpolation, to improve the performance and generalization ability of head gesture recognition models. The results demonstrate that these techniques significantly enhance model performance, particularly for underrepresented gesture classes, by increasing the diversity of training samples and balancing the class distribution.

5.2 Limitations of the Current Approach

This study has several limitations, like the generalization ability of the proposed pipeline across different speaker configurations and gesture classes is limited, as evidenced by the group k-fold cross-validation analysis, consistent with the findings of previous studies (Yang, 2022; Lyu, 2023). The robustness of the system to variations in gesture speed, duration, and execution across individuals requires further investigation, a limitation shared by other studies (Lin, 2021; Wang, 2023).

The current approach does not explicitly address the distinction between intentional head gestures and unintentional head movements during natural conversations, a common limitation among existing studies (Onuonga, 2023). The computational complexity and memory requirements of the HydraMultiROCKET model may pose challenges for real-time deployment on resource-constrained devices. However, the superior performance of HydraMultiROCKET justifies its use in our pipeline as our main objective was performance and not for real-time detection.

Furthermore, our study relies on a dataset that may not fully capture the diversity of head gestures across different cultures, ages, and conversational contexts, a limitation shared by most previous works in the field (Wang, 2023; Onuonga, 2023). Addressing these limitations in future research is crucial to develop more accurate, adaptable, and practically applicable head gesture recognition systems.

5.3 Future Research Directions

Based on the findings and limitations of this study, several future research directions are proposed:

1. Explore the use of multi-modal methods like in (van Amsterdam et al., 2022) where both kinematic and video data was used in conjunction. Since there is access to speaker video recording, there is a possibility of improvement in performance by allowing the model to dynamically weigh different input modalities (e.g., visual and kinematic data), leading to more accurate and robust predictions.

2. Investigate the impact of rotation vector normalization on the accuracy of head gesture recognition compared to other representation methods like Euler angles or quaternions like in this paper (Hachaj and Piekarczyk, 2019).

3. Analyze the performance of the proposed pipeline across different types of conversations (e.g., formal vs informal, emotional vs neutral) and speaker personalities to assess its adaptability and robustness in various contexts.

4. Explore the potential of transfer learning by applying the trained models to other motion capture datasets to assess their generalization ability and adaptability to different data sources, which could potentially extend the applicability of the proposed approach to a wider range of scenarios.

5. Expand the current dataset by annotating more recordings, which could potentially improve the performance and robustness of the head gesture recognition system by providing a more diverse and representative set of examples for training and evaluation.

6. Develop techniques to distinguish between intentional head gestures and unintentional head movements during natural conversations, which could enhance the practical applicability of the system in real-world settings.

7. Explore the potential of unsupervised or self-supervised pre-training on unlabeled motion capture data to improve the overall performance of the gesture recognition pipeline, as this approach could leverage the abundance of unlabeled data to learn meaningful representations and reduce the reliance on manually annotated examples.

Bibliography

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blagus, R. and Lusa, L. (2013). Smote for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 14(1).

Bogduk, N. and Mercer, S. (2000). Biomechanics of the cervical spine. i: Normal kinematics. *Clinical Biomechanics*.

Chen, X. (2023). Automatic classification of human head gestures using motion capture data. Master's thesis, University of Edinburgh. Master's thesis, University of Edinburgh, School of Informatics.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Dempster, A., Petitjean, F., and Webb, G. I. (2020). Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495.

Dempster, A., Schmidt, D. F., and Webb, G. I. (2021). Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery amp; Data Mining*, KDD '21. ACM.

Dempster, A., Schmidt, D. F., and Webb, G. I. (2022). Hydra: Competing convolutional kernels for fast and accurate time series classification.

Diebel, J. et al. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35.

Ephrat, A., Mosseri, I., Lang, O., Dekel, T., Wilson, K., Hassidim, A., Freeman, W. T., and Rubinstein, M. (2018). Looking to listen at the cocktail party: a speaker-

independent audio-visual model for speech separation. *ACM Transactions on Graphics*, 37(4):1–11.

Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., and Petitjean, F. (2019). Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962.

Foumani, N. M., Tan, C. W., Webb, G. I., and Salehi, M. (2023). Improving position encoding of transformers for multivariate time series classification. *Data Mining and Knowledge Discovery*, 38(1):22–48.

Gong, W., Fang, S., Yang, G., and Ge, M. (2017). Using a hidden markov model for improving the spatial-temporal consistency of time series land cover classification. *ISPRS International Journal of Geo-Information*, 6(10).

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Graves, A. (2012). Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376. ACM.

Haag, K. and Shimodaira, H. (2015). The university of edinburgh speaker personality and mocap dataset. In *Proceedings of the 17th ACM International Conference on Multimodal Interaction*, pages 437–444.

Hachaj, T. and Piekarczyk, M. (2019). Evaluation of pattern recognition methods for head gesture-based interface of a virtual reality helmet equipped with a single imu sensor. *Sensors*, 19(24).

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science Business Media.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hu, W. and Zhao, S. (2022). Remaining useful life prediction of lithium-ion batteries based on wavelet denoising and transformer neural network. *Frontiers in Energy Research*, 10:969168.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963.

Iwana, B. K. and Uchida, S. (2021). An empirical survey of data augmentation for time series classification with neural networks. *PLOS ONE*, 16(7):e0254841.

Jaén-Vargas, M., Reyes Leiva, K. M., Fernandes, F., Barroso Gonçalves, S., Tavares Silva, M., Lopes, D. S., and Serrano Olmedo, J. J. (2022). Effects of sliding window variation in the performance of acceleration-based human activity recognition using deep learning models. *PeerJ Comput Sci*, 8:e1052. ©2022 Jaén-Vargas et al.

Johnson, J. M. and Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27.

Karim, F., Majumdar, S., Darabi, H., and Chen, S. (2018). Lstm fully convolutional networks for time series classification. *IEEE Access*, 6:1662–1669.

Khan, A. A., Chaudhari, O., and Chandra, R. (2024). A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation. *Expert Systems with Applications*, 244:122778.

Kukačka, J., Golkov, V., and Cremers, D. (2017). Regularization for deep learning: A taxonomy.

Köpüklü, O., Gunduz, A., Kose, N., and Rigoll, G. (2019). Real-time hand gesture detection and classification using convolutional neural networks.

Liang, Y., Wen, H., Nie, Y., Jiang, Y., Jin, M., Song, D., Pan, S., and Wen, Q. (2024). Foundation Models for Time Series Analysis: A tutorial and survey. *arXiv.org*.

Lin, T. (2021). Automatic classification and segmentation of human head gestures. Master's thesis, University of Edinburgh.

Lyu, Z. (2023). Automatic classification of human head gestures with neural networks. Master's thesis, University of Edinburgh.

Middlehurst, M., Schäfer, P., and Bagnall, A. (2024). Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 38(4):1958–2031.

Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324.

Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., and Kautz, J. (2016). Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4207–4215. IEEE.

Núñez, J. C., Cabido, R., Pantrigo, J. J., Montemayor, A. S., and Vélez, J. F. (2018). Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. *Pattern Recognition*, 76:80–94.

Onuonga, E. (2023). Isolated head gesture recognition using advanced neural networks. Master's thesis, University of Edinburgh.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). *GroupKFold*.

Pérez, S. N. C. and Borz, S. A. (2021). Improving the event-based classification accuracy in pit-drilling operations: An application by neural networks and median filtering of the acceleration input signal data. *Sensors (Basel)*, 21(18):6288.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Romain Ilbert, Thai V. Hoang, Z. Z. (2024). Data augmentation for multivariate time series classification: An experimental study.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Van Ess-Dykema, C., and Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Tan, C. W., Dempster, A., Bergmeir, C., and Webb, G. I. (2022). Multirocket: Multiple pooling operators and transformations for fast and effective time series classification.

van Amsterdam, B., Funke, I., Edwards, E., Speidel, S., Collins, J., Sridhar, A., Kelly, J., Clarkson, M. J., and Stoyanov, D. (2022). Gesture recognition in robotic surgery with multimodal attention. *IEEE Transactions on Medical Imaging*, 41(7):1677–1687. Epub 2022 Jun 30.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

Wang, K., Li, K., Zhou, L., Hu, Y., Cheng, Z., Liu, J., and Chen, C. (2019). Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing*, 360:107–119.

Wang, Y. (2023). Enhancing head gesture recognition using cnn-lstm hybrid models. Master's thesis, University of Edinburgh.

Yang, Y., Shum, H. P. H., Aslam, N., and Zeng, L. (2016). Temporal clustering of motion capture data with optimal partitioning. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, VRCAI '16, page 479–482, New York, NY, USA. Association for Computing Machinery.

Yang, Z. (2022). Automatic classification and clustering of human head gestures. Master's thesis, University of Edinburgh.

Yao, A. and Fu, J. (2014). Deformable gesture recognition. *Computer Vision and Image Understanding*, 118:99–110.

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., and Eickhoff, C. (2020). A transformer-based framework for multivariate time series representation learning.

Zhang, A., Zhou, P., Huang, K., Zou, Y., Liu, M., and Xie, L. (2023). U2-kws: Unified two-pass open-vocabulary keyword spotting with keyword bias.

Zhang, G., Wang, C., Xu, B., and Grosse, R. (2018). Three mechanisms of weight decay regularization.

Appendix A

First appendix

A.1 Sliding Window Experiment Result

Stride	Accuracy	F1-Score	Computational Time (s)
2	0.868	0.852	580
6	0.843	0.829	210
10	0.825	0.811	62
15	0.807	0.793	45

Table A.1: Performance comparison of different stride lengths

A.2 TST architecture

A.3 Hyper parameter Tuning results

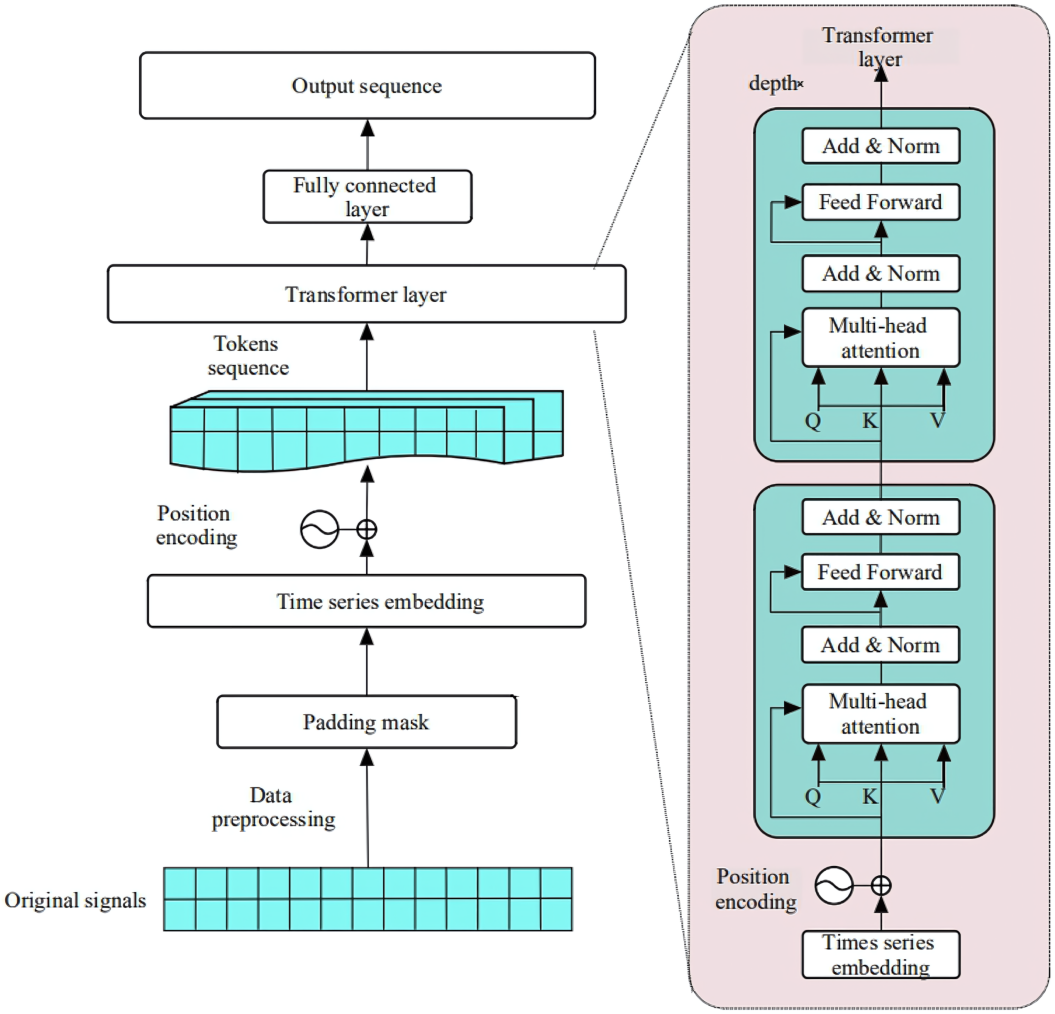

Figure A.1: Time Series Transformer Architecture from (Hu and Zhao, 2022)

fc_dropout	Weight Decay	F1-Score	Accuracy	ROC AUC
0.5	0.01	0.832	0.830	0.901
0.5	0.1	0.838	0.836	0.907
0.5	0.001	0.829	0.827	0.898
0.8	0.1	**0.844**	**0.842**	**0.912**
0.8	0.01	0.840	0.838	0.909
0.8	0.001	0.835	0.833	0.904
0.9	0.1	0.841	0.839	0.910
0.9	0.01	0.837	0.835	0.906
0.9	0.001	0.833	0.831	0.902

Table A.2: Hyperparameter tuning results for Binary Gesture Detection HydraMultiROCKET

fc_dropout	Weight Decay	F1-Score	Accuracy	ROC AUC
0.5	0.01	0.620	0.640	0.780
0.5	0.1	0.625	0.645	0.785
0.5	0.001	0.615	0.635	0.775
0.8	0.1	0.630	0.650	0.790
0.8	0.01	0.628	0.648	0.788
0.8	0.001	0.622	0.642	0.782
0.9	0.1	0.629	0.649	0.789
0.9	0.01	0.627	0.647	0.787
0.9	0.001	0.621	0.641	0.781

Table A.3: Hyperparameter tuning results for HydraMultiROCKET for Gesture Classification