Democratic and Popular Algerian Republic

Ministry of Higher Education and Scientific Research

Yahia fares university of Médéa



Faculty of Sciences

Department of Mathematics and Computer Science

Final Master Project

Branch: Computer Science

Specialty: Systems Engineering and web Technologies

Theme

Deep Learning for Word Boundary Predictor in Continuous Sign Language

Prepared and presented by

BETTACHE INNAS
MAI SOULAF

Proposed and directed by

Dr. KHELDOUN AHMED

***Defended on June , 2024, In front of the jury composed of*:**

Dr. Mohamed Yassine Hayi: Yahia Fares University - President

Mr. Ismail Hadjir: Yahia Fares University - Examiner

Year: 2023-2024

ACKNOWLEDGMENTS

First and foremost, we thank God for His guidance and blessings throughout our academic journey and the completion of this thesis.

We extend our deepest gratitude to our supervisor, Dr. Kheldoun, for his invaluable teaching, guidance, and wise advice. His support has been instrumental in the completion of this work.

Our thanks go to Dr. Hayi for graciously agreeing to chair the commission for evaluating our work. His participation and insights are greatly appreciated.

We also thank Mr. Hadjir for agreeing to examine our work. His critical evaluation and feedback have been crucial in refining our research.

Special thanks to Ms. Louz for her assistance throughout this process. Her help has been greatly valued.

Our heartfelt thanks also go to all our professors for their generosity and great patience. Despite their academic and professional responsibilities, they have shown immense support and dedication to our growth and learning.

Dedication

First and foremost, Alhamdulillah, all praise and thanks are due to ***Allah*** *(SWT) for giving us power and patience to complete this work. I dedicate this thesis to all the people who have been by my side this thesis is the fruit of our collective efforts, and I am honored to dedicate it to each one of you.*

To my dear ***mother****, your constant encouragement and unwavering faith in me have allowed me to continually excel. Your dedication and prayers have been my beacon in the darkest moments. Each of my successes is a tribute to your love and countless sacrifices.*

To my dear ***father****, your wisdom and determination have shaped the person I am today. Your sacrifices have been the pillars that allowed me to reach this important moment in my life.*

To my sister ***Dounia*** *and my brothers* ***Noureddine*** *and* ***Ayad****, your unwavering support and unconditional love have been invaluable. May God protect you and grant you happiness.*

To my cousins ***Soumia*** *and* ***Chaima****, more than family, you are my sisters.Your presence in both good and bad times, along with your unwavering support and love, has been invaluable. I thank you for brightening my life.*

To my dear friends ***Marwa*** *and* ***Amira****, your support and presence are invaluable, you have been much more than companions in life. Your discussions, exchanges, and mutual aid have been sources of inspiration and motivation throughout this journey. Thank you for sharing this adventure with me and making these years of study so memorable.*

To you, my partner ***Innas****, I wish to express my gratitude for your constant presence, valuable collaboration, unwavering support, and understanding throughout this project. We formed a strong and complementary team, and our joint work was essential to the completion of this project.*

To my professor and supervisor, ***Dr. Kheldoun****, I wish to express my gratitude for your teaching, guidance, and wise advice. I am honored to have had the chance to benefit from your support throughout this thesis.*

Madame Louz Khadija*, I am truly grateful for your help and the facilitation you provided during this journey. Your efforts in making this journey smooth and successful are highly valued and acknowledged.*

I am deeply grateful to all those who have been by my side. Thank you for your help and support during this journey.

Finally, I dedicate this thesis to ***myself****, in recognition of my commitment, determination, and perseverance. This work is a reflection of my investment and passion for my studies. I am proud of this achievement and grateful to myself for pushing my limits and reaching my goals.*

Mai Soulaf

Dedication

Alhamdulillah, all praise and thanks are due to Allah (SWT) for giving us the power and patience during this work accomplishment.

Dad, this is for you. No words can capture the depth of my gratitude for your belief in me. Your constant support fueled my determination through countless late nights and moments of crushing doubt. When fear threatened to derail me, you were my lighthouse, guiding me back on course. Thank you for being the rock I could always lean on, the safe haven where I found understanding and love, exactly when I needed them most.

Mom, this wouldn't be here without you. Your constant love and understanding were the warm embrace that kept me going. You knew exactly when a word of encouragement or a quiet moment of comfort was needed. Thank you for being my anchor, the steady presence that grounded me through every challenge. Your support is a gift I'll forever cherish.

To my incredible siblings, **Dahman and Djihad**, this journey wouldn't have been the same without your belief in me. You weren't just siblings, you were my personal cheerleaders, celebrating every milestone and pushing me forward even when I stumbled. Thank you for the countless inside jokes that kept me laughing, the late-night study sessions that fueled my focus, and the support that reminded me I wasn't alone. This thesis is a testament to the strength of our bond, and I couldn't have achieved it without you by my side.

To **Madame Louz Khadija**, thank you for your help and the facilitation you provided throughout this journey. It is greatly appreciated.

To my supervisor, **Khaldoun Ahmed**, your guidance and support throughout this thesis journey have been invaluable. I am grateful for your expertise and dedication, which have helped me achieve this milestone.

To my binome, **Mai Soulaf**, your friendship and collaboration made this journey not just productive but enjoyable. Together, we achieved more than I could have imagined..

To the closest to my heart **Nesrine Belaid** and **Raouf Benyahia**, this thesis wouldn't be the same without your incredible support. Our brainstorming sessions weren't just about work, they were filled with laughter that kept me going and even a few tears we shared through the challenges. Thank you for being my sounding boards, my cheerleaders. Your presence by my side, filled with laughter and support, not only fueled my determination but also made this journey an unforgettable chapter in my life.

To **my future self**, may this work be a reminder of what I am capable of achieving with passion and effort.

Bettache Innas

Abstract

This document presents the development and evaluation of BPredictor4CSL, a mobile app designed to predict sign language word boundaries in continuous signing videos. By leveraging hybrid learning techniques, particularly Bidirectional Long Short-Term Memory (BiLSTM) networks, along with machine learning models like Random Forest (RF) and Gradient Boosting (GB), BPredictor4CSL aims to enhance the accuracy and efficiency of automated sign language word boundary detection. The system utilizes the DGS corpus as its dataset, ensuring a comprehensive representation of sign language. The objective is to facilitate communication between the deaf community and non-deaf individuals. Our research includes comprehensive data preparation, preprocessing, model training, and evaluation phases. Key steps involve video frame extraction, facial and hand landmark detection, and feature extraction, culminating in a robust dataset for training the predictive model. The system's performance is assessed using metrics such as confusion matrices, classification reports, AUPRC, and ROC AUC scores, demonstrating its potential to significantly improve communication for those reliant on sign language.

Keywords: sign language, deep learning, BiLSTM networks, machine learning, random forests, gradient boosting, facial landmark detection, hand landmark detection, feature extraction, communication facilitation, continuous sign language videos, predictive model evaluation,hybrid learning.

Resumé

Ce document présente le développement et l'évaluation de BPredictor4CSL, une application mobile conçue pour prédire les frontières des mots en langue des signes dans des vidéos de signes continus. En utilisant des techniques d'apprentissage hybride, en particulier les réseaux de mémoire à long terme bidirectionnels (BiLSTM), ainsi que des modèles d'apprentissage automatique comme Random Forest (RF) et Gradient Boosting (GB), BPredictor4CSL vise à améliorer la précision et l'efficacité de la détection automatique des frontières des mots en langue des signes. Le système utilise le corpus DGS comme ensemble de données, garantissant une représentation complète de la langue des signes. L'objectif est de faciliter la communication entre la communauté des sourds et les entendants. Notre recherche inclut une préparation complète des données, un prétraitement, une formation et des phases d'évaluation des modèles. Les étapes clés comprennent l'extraction des images vidéo, la détection des points de repère du visage et des mains, et l'extraction des caractéristiques, aboutissant à un ensemble de données robuste pour la formation du modèle prédictif. Les performances du système sont évaluées à l'aide de métriques telles que les matrices de confusion, les rapports de classification, l'AUPRC et les scores ROC AUC, démontrant son potentiel à améliorer significativement la communication pour ceux qui dépendent de la langue des signes.

Mots-clés : langue des signes, apprentissage profond, réseaux BiLSTM, apprentissage automatique, forêts aléatoires, gradient boosting, détection des points de repère du visage, détection des points de repère des mains, extraction des caractéristiques, facilitation de la communication, vidéos en langue des signes continues, évaluation du modèle prédictif, apprentissage hybride.

ملخص

يهدف هذا العمل الى تطوير وتقييم نظام CSL4BPredictor وهو تطبيق جوال مصمم للتنبؤ بحدود الكلمات في لغة الاشارة وهذا من خلال مقاطع فيديوا. بالاستعانة بتقنيات التعلم الهجين, وخاصة شبكات الذاكرة طويلة وقصيرة المدى ثنائية الاتجاه ،(BiLSTM) بالإضافة إلى نماذج التعلم الآلي مثل الغابة العشوائية (RF) وتعزيز التدرج (GB) ، يجعل BPredictor4CSL ذو دقة وكفاءة في الكشف التلقائي عن حدود الكلمات في لغة الإشارة. يستخدم النظام مجموعة بيانات DGS لضمان تمثيل شامل للغة الإشارة. الهدف هو تسهيل التواصل بين مجتمع الصم والأفراد غير الصم. تشمل أبحاثنا إعداد البيانات بشكل شامل، والمعالجة المسبقة، وتدريب النماذج، ومراحل التقييم. تتضمن الخطوات الرئيسية استخراج إطارات الفيديو، واكتشاف معالم الوجه واليد، واستخراج الميزات، مما يؤدي إلى مجموعة بيانات قوية لتدريب النموذج التنبؤي. يتم تقييم أداء النظام باستخدام مقاييس مثل مصفوفات الارتباك، وتقارير التصنيف، و ،AUPRC و AUC ROC ، مما يدل على إمكاناته في تحسين التواصل بشكل كبير لأولئك الذين يعتمدون على لغة الإشارة.

الكلمات المفتاحية : لغة الإشارة، التعلم العميق، شبكات BiLSTM ، التعلم الآلي، الغابات العشوائية، تعزيز التدرج، اكتشاف معالم الوجه، اكتشاف معالم اليد، استخراج الميزات، إتاحة التواصل، فيديوهات لغة الإشارة المستمرة، تقييم النموذج التنبؤي، التعلم الهجين.

Contents

Table of Contents i

List of Tables iii

List of Figures v

List of Abbreviations vi

General Introduction 1

1 Sign Language Overview 4

1.1 Introduction . . . 5
1.2 Overview on sign language . . . 5
1.3 Linguistic Features of Sign Languages . . . 6
1.4 Algerian Sign Language . . . 9
1.5 Isolated Sign Language and Continuous Sign Language . . . 10
1.5.1 Isolated Sign Language . . . 10
1.5.2 Continuous Sign Language . . . 11
1.5.3 Differences between Isolated and Continuous Sign Language . . . 11
1.6 Challenges and Solutions in Continuous Sign Language Recognition . . . 11
1.7 Word Boundary Prediction for CSL . . . 12
1.8 Conclusion . . . 12

2 Deep Learning Overview 14

2.1 Introduction . . . 15
2.2 Artificial intelligence . . . 15
2.2.1 Machine Learning . . . 16
2.2.2 Deep Learning . . . 17
2.3 Artificial Neural Networks . . . 17
2.3.1 The neural network layers . . . 18
2.3.2 Weight . . . 18
2.3.3 Bias . . . 18
2.3.4 Neurons . . . 19
2.3.5 Activation function . . . 19
2.3.6 Loss function . . . 20

2.4 Convolutional Neural Network(CNN) 21
2.5 Recurrent Neural Networks 23
2.6 Long Short-Term Memory networks 23
2.7 Autoencoder 24
2.8 Evaluation metrics of models 26
2.9 Related Works 28
2.10 Conclusion 29

3 Design of the BPredictor4CSL system 30
3.1 Introduction 31
3.2 Problem Formulation 31
3.3 Overview of the BPredictor4CSL system 33
3.4 Pre-Processing 34
3.5 Proposed Model 40
3.6 Model Architecture 40
3.6.1 Bidirectional LSTM (BiLSTM) 41
3.6.2 Random Forest Classifier Training 42
3.6.3 Gradient Boosting Classifier Training 43
3.6.4 Combining Predictions 44
3.7 Conclusion 45

4 Results and Discussion 47
4.1 Introduction 48
4.2 Development environment 48
4.2.1 Hardware specifications 48
4.2.2 Software (Python) 49
4.2.3 Libraries 49
4.3 Dataset (DGS corpus) 50
4.4 Results 50
4.4.1 Hyperparameters 51
4.4.2 Performance Metrics of Models 54
4.4.3 Comparison Results(Test Models on DGS corpus Dataset 57
4.5 Comparison Results with Model Proposed in Literature 59
4.6 Developed Tools: BPredictor4CSL 61
4.6.1 Application Interface 61
4.6.2 Key Functionalities 64
4.7 Conclusion 64

General Conclusion 66

Bibliography 68

List of Tables

1.1 ASL vs ALG-SL 6
1.2 Differences between Isolated and Continuous Sign Language [30, 31]. 11

2.1 Comparison of Machine Learning Methods 17
2.2 Summary of Activation Functions 20

3.1 Number of Landmarks for Facial and Hand Landmarks 36
3.2 A sample of created dataset before cleaning 39
3.3 A sample of created dataset after cleaning 39

4.1 Hyperparameters of the BiLSTM Model 52
4.2 Hyperparameters of the Random Forest Model 53
4.3 Hyperparameters of the Gradient Boosting Model 53
4.4 Classification Report for BiLSTM, Random Forest, and Gradient Boosting Models 55
4.5 Confusion Matrix for BiLSTM, Random Forest, Gradient Boosting, and Combined Models 57
4.6 Performance Evaluation of Various Models on DGS Dataset 58
4.7 Performance Comparison of DGS dataset 60

List of Figures

Figure 1.1 Various handshapes and their detailed descriptions of base knuckle and interphalangeal articulation. 7
Figure 1.2 Minimal contrasts of signs illustrating major parameters 7
Figure 1.3 Sign of the letter 'V' and the word 'two' 8
Figure 1.4 Facial expresion 8
Figure 1.5 Algerian Sign Language Alphabet (the ones with red border are the dynamic signs) 10
Figure 1.6 Conceptual illustration of word boundary prediction in continuous sign language using deep learning. 12

Figure 2.1 Visualizing Components of Artificial Intelligence 15
Figure 2.2 Artifitial Neural Network architecture 18
Figure 2.3 Simple perceptron. 19
Figure 2.4 Sample convolutional neural network architecture for image classification 21
Figure 2.5 The convolution process 22
Figure 2.6 The process of different pooling layer 22
Figure 2.7 Architecture of Recurrent Neural Networks 23
Figure 2.8 LSTM basic unit 24
Figure 2.9 Autoencoder architecture 25
Figure 2.10 Confusion matrix Table structure 28

Figure 3.1 BPredictor4CSL: Predicting Boundary Labels for Continuous Sign Language 32
Figure 3.2 General diagram of the BPredictor4CSL 33
Figure 3.3 Pipeline for Feature Extraction from Video Frames 37
Figure 3.4 Sample Label Vector 38
Figure 3.5 Detailed pre-processing pipeline for video data 40
Figure 3.6 Detailed structure of the BiLSTM model, including input shape and output shape of each layer. 42
Figure 3.7 Data preparation for Random Forest and Gradient Boosting Classifiers training. 44
Figure 3.8 Ensemble Classifier Workflow: Combining Predictions from Random Forest, Gradient Boosting, and BiLSTM Models. 45

Figure 4.1 Graphical interfaces of the input 62
Figure 4.2 Graphical interface of the output 63

List of Abbreviations

AE Autoencoders

AI Artificial intelligence

ALGSL Algerian Sign Language

ANNs Artificial Neural Networks

ASL American Sign Language

AUPRC The Area Under the Precision-Recall Curve

AU-ROC Area Under the Receiver Operating Characteristic Curve

BiLSTM Bidirectional Long Short-Term Memory

CNN Convolutional Neural Network

CRFs conditional random fields

CSL Continuous Sign Language

CSLR Continuous Sign Language Recognition

DGS Deutsche Gebärdensprache (German sign language)

DL Deep Learning

FN False Negatives

FP False Positives

GB Gradient Boosting

GSL German Sign Language

HMM hidden Markov models

ISL Isolated Sign Language

LSF French sign language

LSTM Long Short-Term Memory

ML Machine Learning

NLP Natural Language Processing

NaN Not a Number

RBG Red, Green and Blue

RF Random Forest

RNN Recurrent Neural Network

SVM Support Vector Machine

TN True Negatives

TP True Positives

General Introduction

Sign language plays a pivotal role as a primary mode of communication for the deaf community, facilitating rich and nuanced interactions that extend beyond mere gestures. It constitutes a sophisticated system with its own grammar, syntax, and semantic nuances, akin to spoken languages. Continuous sign language, in particular, presents a formidable challenge due to its dynamic nature, characterized by fluid sequences of movements and expressions. This fluidity makes accurate interpretation inherently complex, requiring sophisticated computational models to decipher and delineate meaningful linguistic units.

Among the myriad tasks in sign language processing, predicting word boundaries in continuous sign language stands out as both exceptionally challenging and crucial. Accurate segmentation of sign streams into discrete lexical units is essential for enabling effective communication and comprehension. Without precise boundary detection, the ability to convey meaning and intent through sign language may be compromised, hindering the integration and participation of deaf individuals in various societal contexts.

This research addresses the urgent need for enhanced communication tools that can bridge the gap between the deaf community and mainstream society. The accurate prediction of word boundaries in continuous sign language holds the potential to revolutionize real-time translation systems, educational resources, and accessibility applications. By enabling more accurate and efficient interpretation of sign language, these advancements aim to foster inclusivity and equitable access to information for all individuals, regardless of their hearing abilities.

The motivation for this study is driven by dual imperatives. Firstly, there is a compelling need to overcome existing communication barriers faced by deaf individuals. Current technologies often struggle to capture the intricacies and nuances of continuous sign language, leading to inaccurate interpretations and misunderstandings. Accurately predicting word boundaries represents a foundational step towards developing robust recognition systems capable of facilitating seamless interactions between deaf and hearing individuals.

Secondly, the rapid advancements in artificial intelligence (AI) and machine learning (ML) offer promising avenues to address the complex challenges associated with predicting word boundaries in continuous sign language. Traditional approaches frequently fall short in handling the dynamic and context-dependent nature of sign language. Leveraging advanced techniques such as BiLSTM networks, Random Forests, and Gradient Boosting classifiers, along with adopting a hybrid learning approach, this research endeavors to expand the frontiers of sign language processing capabilities.

The primary objective of this research is to design and implement BPredictor4CSL, a mobile application specifically tailored to accurately predict word boundaries in continuous sign language videos. This entails developing a robust prediction system capable of effectively identifying and delineating word boundaries amidst the fluid and dynamic nature of sign language streams. By integrating hybrid learning techniques , including BiLSTM networks, Random Forests, and Gradient Boosting classifiers, the aim is to significantly enhance the performance, accuracy, and robustness of the prediction system.

Comprehensive evaluation using DGS Corpus dataset and rigorous performance metrics ensures the reliability and applicability of the BPredictor4CSL system in real-world scenarios. The findings from this research are expected to contribute valuable insights and innovations to the field of sign language processing, with implications extending to areas such as assistive technology development, linguistic research, and inclusive communication strategies.

Research Questions

In this work, we will try to answer the following questions:

1. How effectively can hybrid learning techniques improve the accuracy of word boundary prediction in continuous sign language?
2. What are the key challenges in predicting word boundaries in continuous sign language, and how can they be addressed using advanced AI and ML techniques?
3. How reliable and applicable is the BPredictor4CSL system in real-world scenarios?
4. What impact can accurate word boundary prediction have on the development of assistive technologies and inclusive communication strategies?
5. How does the integration of multiple machine learning models contribute to the overall performance and robustness of the BPredictor4CSL system?

This report is structured into four chapters:

- **Chapter 1: Sign Language Overview:** Introduces sign language, emphasizing the distinction between isolated and continuous forms, and discusses linguistic features and inherent recognition challenges.
- **Chapter 2: Deep Learning:** Provides an overview of deep learning methodologies, including artificial neural networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and Long Short-Term Memory (LSTM) networks. It reviews relevant literature and discusses various model evaluation metrics.

- **Chapter 3: Design of the BPredictor4CSL System:** Outlines the problem formulation, system architecture, data preprocessing techniques, and the proposed machine learning model, emphasizing the use of hybrid learning.
- **Chapter 4: Results and Discussion:** Presents a comprehensive analysis of experimental results, including performance metrics, hyperparameter tuning, and comparisons with existing models. It highlights the effectiveness and limitations of the BPredictor4CSL system.

Chapter 1

Sign Language Overview

1.1 Introduction

Sign languages are rich visual-gestural communication systems that are used by deaf communities around the world. This chapter provides an overview of sign languages, focusing on their linguistic features, regional variations, and the distinction between isolated and continuous sign language. It also highlights the challenges and solutions in continuous sign language recognition, with a particular emphasis on word boundary prediction The chapter begins by introducing sign languages and their unique modality, which uses manual gestures, facial expressions, and body movements to convey meaning. It discusses the essential linguistic parameters, such as handshapes, locations, orientations, and facial expressions, that contribute to the formation of signs. The chapter then delves into the Algerian Sign Language (ALGSL), exploring its origins, official recognition, and regional variations. It provides a comprehensive overview of the ALGSL alphabet, comprising both static and dynamic signs, and highlights the importance of understanding the language's gestural system.

1.2 Overview on sign language

Sign language is the natural language of deaf people. It is a non-verbal and visual language, different in form from spoken language but serving the same function. Spread throughout the world, it is not a universal language. Regionally different languages have evolved, such as GSL (German Sign Language) in Germany or ASL (American Sign Language) in the United States [28]. Contrary to a common misconception, sign language is not universal. There exists a significant diversity of sign languages globally, each with its own unique set of linguistic rules, grammar, and vocabulary. This diversity reflects the rich tapestry of Deaf cultures and communities around the world. For instance, American Sign Language (ASL) and British Sign Language (BSL) are two distinct sign languages used in countries where English is the predominant spoken language. Despite this shared spoken language, ASL and BSL differ substantially from each other, underscoring the fact that sign languages are not merely manual representations of spoken languages but are natural languages in their own right [43] [44] . The development and evolution of sign languages have been influenced by various factors, including geographical, social, and cultural contexts. Studies have shown that sign languages can evolve independently within different communities of the Deaf, leading to the formation of distinct linguistic systems [45] . For example, the evolutionary dynamics in the dispersal of sign languages have revealed that contemporary sign languages can be grouped into several main European lineages, indicating a complex history of development and divergence [45]. Furthermore, the acquisition of sign languages by children mirrors the process of acquiring spoken languages, with infants sensitive to the visual information provided. This process is crucial for the cognitive and social development of Deaf children, highlighting the importance of early exposure to sign language [46] . The diversity of sign languages is further evidenced by research focusing on specific linguistic aspects, such as phonology, which examines the parameters shaping a sign, including hand shape, movement, location, orientation, and facial

expression [47]. These studies contribute to our understanding of the universality and variability within sign language phonology, offering insights into the complex structure and function of sign languages. Table 1.1 provides a visual comparison of handshapes for specific letters between ASL and ALG-SL, illustrating one aspect of this diversity.

In conclusion, the notion of a universal sign language is a misconception. The world's Deaf communities utilize more than 300 documented sign languages, each with its own unique linguistic identity. This diversity reflects the rich cultural and linguistic heritage of Deaf people worldwide, emphasizing the need for recognition and support of sign languages in their respective sociolinguistic contexts [43] [44] [45] .

Table 1.1: ASL vs ALG-SL

Handshape	ASL	ALG-SL
	U	ث
	S	ص
	H	ع
	Y	ي
	I	م

1.3 Linguistic Features of Sign Languages

Sign languages employ manual gestures, facial expressions, and body movements to communicate, forming a visual-gestural modality of communication [29].

- *Handshapes*: The shape and configuration of the hands convey specific meanings [32].



Figure 1.1: Various handshapes and their detailed descriptions of base knuckle and interphalangeal articulation.

- *Location*: Signs are often performed in specific locations in relation to the body or signing space [32].

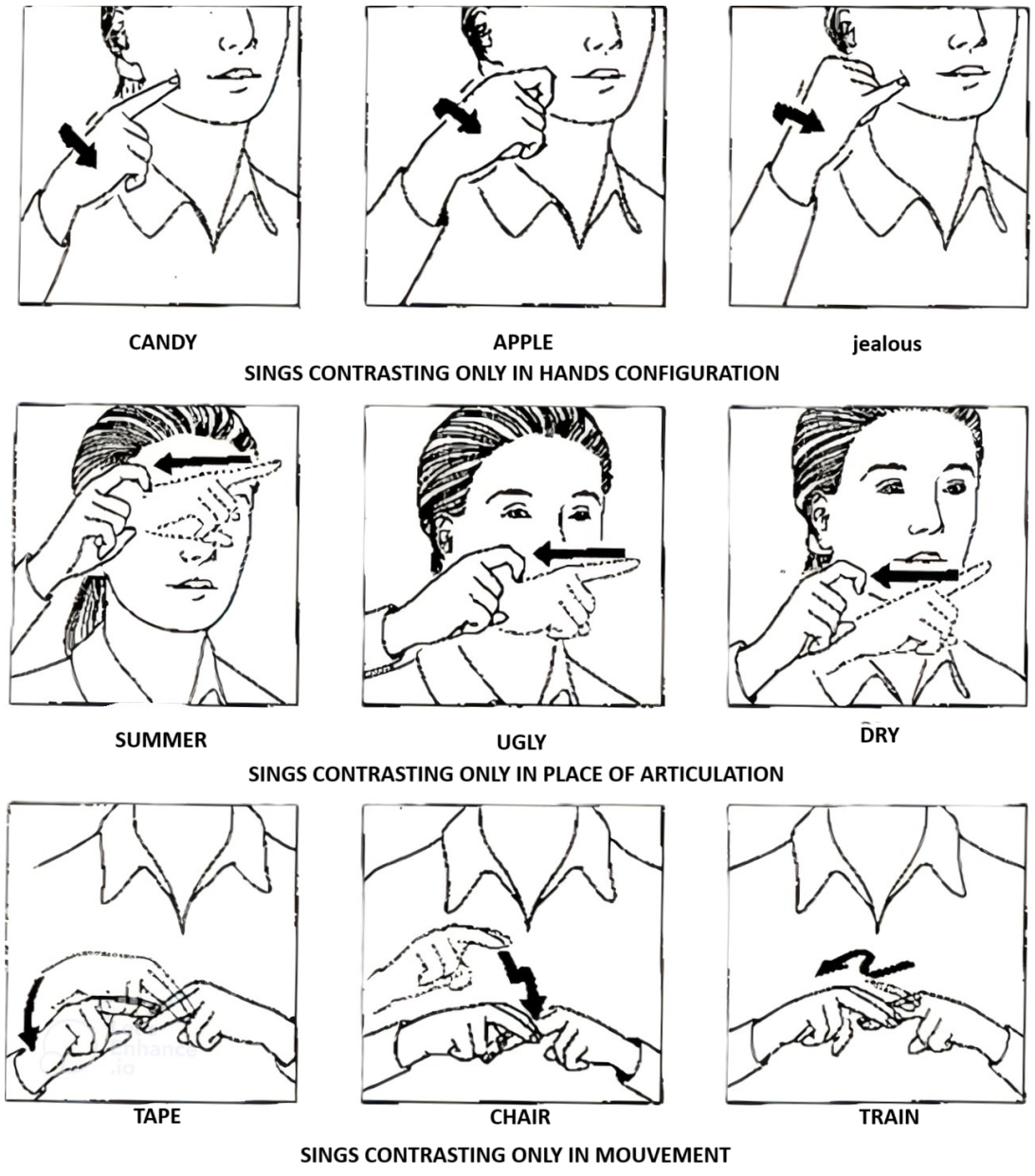


Figure 1.2: Minimal contrasts of signs illustrating major parameters

- *Hand Orientation* : Orientation of the hand represents the position of the palm and the fingers in relation to the signer's body and the position of the hands in relation to each other. The palm can be turned up, down, right, left, towards the signer, away from the signer, or edge towards the signer. The tips of the fingers can be directed up, down, right,left, towards or away from the signer, diagonally, etc. In two-handed gestures, the hands can be symmetrical, parallel to each other, one above, behind, or before the other, the wrists can be crossed, the palms can touch the edges, and so on. Some SL researchers see orientation not as a separate parameter, but as a subcategory of handshape. However, as noted in Figure1.3, the ability of orientation to be a single parameter that identifies the semantics of the sign calls for considering it as a distinct parameter.

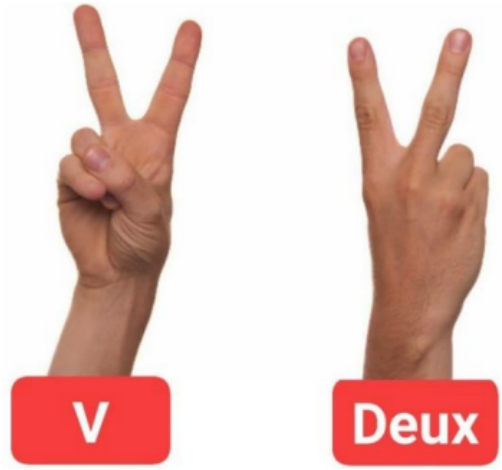


Figure 1.3: Sign of the letter 'V' and the word 'two'

- *Facial expressions*: such as eyebrow raises, lip movements, and smiles provide additional information and emotional context [32].

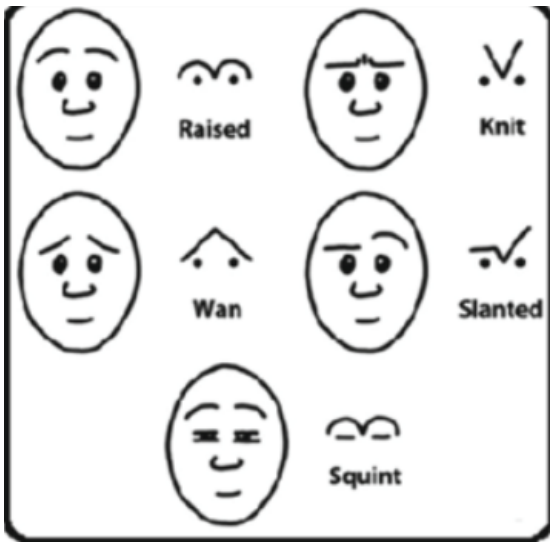


Figure 1.4: Facial expresion

- Body movements such as body posture, head tilts, and shoulder shrugs can contribute to the meaning of a sign or emphasize certain aspects of the message [32].

Sign languages are rich and expressive languages that utilize manual gestures, facial expressions, and body movements to convey meaning effectively.

1.4 Algerian Sign Language

The Algerian Sign Language (ALGSL) is a gestural language used by deaf individuals in Algeria as a means of communication through signs. The ALGSL is officially recognized by the law of 8 May 2002 as the first language of the deaf community in Algeria, making it the only country in the Arab world and in Africa to officially recognize sign language.

The origins of the ALGSL can be traced back to French Sign Language (LSF). It is important to note that Algerian Sign Language is not uniform and comprises various regional variations . These variations include, but are not limited to:

- Algerian Jewish Sign Language (AJSL), also known as Ghardaia Sign Language, which was mainly developed and used in the village of Ghardaia by the Algerian Jewish individuals at that time.
- Algerian Sign Language of Laghouat, used by many Deaf people in Laghouat province and other cities or villages around it.
- Algerian Sign Language of Oran, used by the Deaf in the North of Algeria, particularly in the city of Oran.
- Algerian Sign Language of Adrar, used by the Algerian Deaf community in Adrar, in the South of Algeria.

The ALGSL is a gestural language that is based entirely on signs, which are produced through different parts of the body, including one or both hands, the face, shoulders, or even the entire body. Despite its unique mode of expression, the ALGSL is considered to be a language like any other, possessing a distinct vocabulary and organized syntax. To effectively learn and understand the ALGSL, one must become familiar with its alphabet. In fact, every sign within the ALGSL alphabet is generated by one or two particular hand postures, highlighting the precision and complexity of the language's gestural system.

As can be seen in Figure 1.5, the ALGSL is made up of a total of 42 signs for the alphabet, which include 35 static and 6 dynamic signs. Notably, these signs are produced using a single hand gesture. Moreover, each static sign is identified by two distinct features: configuration and orientation[42].



Figure 1.5: Algerian Sign Language Alphabet (the ones with red border are the dynamic signs)

1.5 Isolated Sign Language and Continuous Sign Language

1.5.1 Isolated Sign Language

Isolated sign language refers to a sign language that has developed in a community of deaf people who have little or no contact with speakers of other languages, either spoken or signed. Isolated sign languages are often very different from other sign languages, as they have developed independently and have not been influenced by other languages. They may have unique grammatical structures, vocabularies, and ways of expressing meaning [30, 31].

Isolated sign languages are characterized by their self-contained nature, with little external influence shaping their development. This independence allows for the emergence of novel signs, grammar rules, and cultural expressions within the community. The linguistic diversity found in isolated sign languages highlights the richness and adaptability of human communication systems in diverse environments.

Researchers have noted that studying isolated sign languages provides valuable insights into the fundamental principles of language evolution and human cognition. By examining these unique linguistic systems, linguists can better understand the universal aspects of language acquisition, processing, and communication strategies across different cultural contexts [30].

1.5.2 Continuous Sign Language

Continuous sign language represents the dynamic flow of sign communication in everyday interactions. Unlike isolated sign languages that rely on discrete signs, continuous sign language emphasizes fluid movements that seamlessly integrate handshapes, facial expressions, and body gestures to convey meaning. This continuous flow of expression enhances the naturalness and expressiveness of signed communication.

Continuous sign languages are characterized by their holistic approach to conveying information, where individual signs are interconnected to form coherent messages. The use of non-manual markers such as facial expressions and body postures plays a crucial role in shaping the nuances and emotional content of continuous signing. This integrated use of multiple channels enhances the clarity and richness of communication in sign language interactions.

Studies have shown that continuous sign language facilitates efficient information processing and comprehension due to its dynamic nature and multimodal features. By exploring the intricate interplay between manual signs and non-manual components in continuous signing, researchers gain valuable insights into the complex mechanisms underlying human communication and cognitive processing [31].

1.5.3 Differences between Isolated and Continuous Sign Language

The following table summarizes the key differences between isolated sign language and continuous sign language:

Feature	**Isolated Sign Language**	**Continuous Sign Language**
Definition	A collection of individual signs that can be produced independently of each other	A fluid stream of movement that conveys meaning through various gestures
Use	Teaching and research purposes	Everyday communication
Naturalness	Less natural	More natural

Table 1.2: Differences between Isolated and Continuous Sign Language [30, 31].

The isolated sign language and the continuous sign language are two different forms of sign language that serve different purposes As shown in Table 1.2. Isolated sign language is useful for teaching and research purposes, while continuous sign language is the natural form of sign language used by deaf people in everyday communication.

1.6 Challenges and Solutions in Continuous Sign Language Recognition

Continuous Sign Language Recognition (CSLR) faces several challenges, including the need for effective segmentation of word gestures in sign language videos [36] , the development of models

that can handle the variability and complexity of sign language [37] [38] [39] , and the integration of various modalities such as hand gestures, facial expressions, and body postures [40] . Solutions to these challenges involve the use of advanced machine learning and deep learning techniques, including reinforcement learning for semantic boundary detection [38] , attention-based models [37] , and the incorporation of skeletal input for sign language segmentation [41] . These approaches aim to improve the accuracy and efficiency of CSLR systems by enhancing their ability to capture and interpret the nuanced features of sign language.

1.7 Word Boundary Prediction for CSL

Word boundary prediction is the task of identifying the boundaries between words in a continuous stream of speech [35]. This is a challenging task, as there are often no clear acoustic cues to indicate where one word ends and another begins. Word boundary prediction is important for various natural language processing tasks, such as speech recognition, machine translation, and text-to-speech synthesis.

There are several approaches to word boundary prediction, including statistical mode such as hidden Markov models (HMMs) or conditional random fields (CRFs) [33], as well as neural network-based approaches [34]. Refer to Figure 1.6 for a conceptual illustration of this process in continuous sign language.

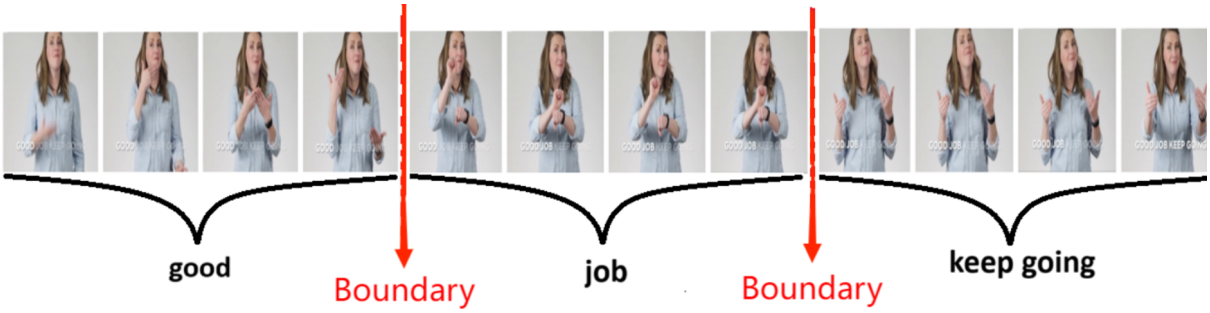


Figure 1.6: Conceptual illustration of word boundary prediction in continuous sign language using deep learning.

1.8 Conclusion

This chapter has provided a comprehensive introduction to sign languages, with a particular focus on the Algerian Sign Language (ALGSL). It has explored the linguistic features of sign languages, including handshapes, locations, orientations, and facial expressions, and their role in communicating meaning effectively.

The chapter has also highlighted the distinction between isolated and continuous sign language, emphasizing the naturalness and fluidity of continuous sign language in everyday communication. In addition, it has discussed the challenges associated with Continuous Sign Language

Recognition (CSLR), such as segmentation, variability, and multimodal integration, and the potential solutions involving advanced machine learning and deep learning techniques.
Furthermore, the chapter has introduced the concept of word boundary prediction, a crucial task in natural language processing, and its application in continuous sign language recognition. It has provided a conceptual illustration of using deep learning for word boundary prediction in continuous sign language.

Chapter 2

Deep Learning Overview

2.1 Introduction

In this chapter, we will offer a comprehensive overview of deep learning, an essential component of artificial intelligence. We will begin by introducing artificial neural networks as the foundational elements of deep learning. We will then delve into advanced deep learning architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Additionally, we will discuss the significance of autoencoders and Long Short-Term Memory (LSTM) networks. After that, we will delve into the evaluation metrics used to assess the performance of these models. Finally, we will conclude with a comprehensive review of previous research in the field, offering insights into the evolution and advancements of deep learning methodologies.

2.2 Artificial intelligence

Artificial Intelligence (AI) is a computer science field that aims to create systems capable of performing tasks that would typically require human cognitive abilities such as learning, reasoning, and problem-solving[9]. AI systems generally work by processing large volumes of labeled training data, analyzing it for correlations and patterns and then using these patterns to predict future states [10]. It is always growing alongside new technologies such as computer vision and natural language processing.

Artificial intelligence is composed of two primary components: machine learning and deep learning. These components are illustrated in Figure 2.1, which visualizes the foundational elements of AI.

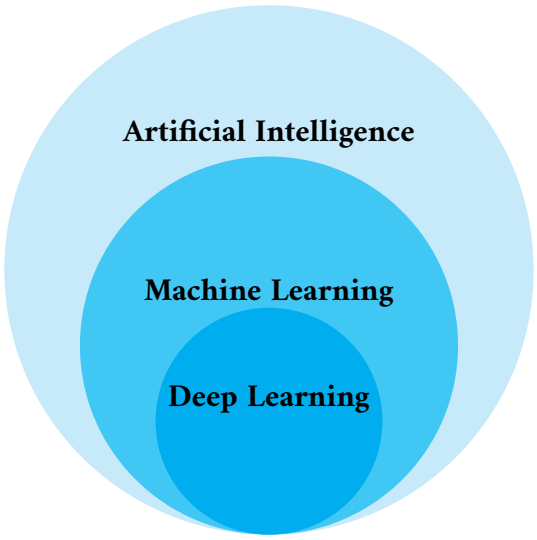


Figure 2.1: Visualizing Components of Artificial Intelligence

2.2.1 Machine Learning

Machine learning (ML) is a form of artificial intelligence (AI) that focuses on creating systems that learn, or improve their performance, based on the data they process [11], without being explicitly programmed .
The most commonly used machine learning methods (algorithms) are supervised, unsupervised, semi-supervised and reinforcement learning. These will be further explained below.

- **Supervised learning:** is a machine learning technique that utilizes labeled datasets, these datasets are used to train or "supervise" algorithms to accurately classify data or predict outcomes, by using labeled inputs and outputs, the model can measure its accuracy and learn over time[12]. It is used to accomplish two main types of tasks: classification and regression.

- **Unsupervised learning :** analyzes and groups unlabeled data sets using machine learning algorithms. Without the need for human assistance, these algorithms find hidden patterns in data[12]. Where it is used in two common applications: clustering and association.

  Unsupervised learning is more challenging to implement and is not as widely used as supervised learning.

- **Semi-supervised learning:** is the hybrid approach in machine learning that combines labeled and unlabeled datasets during the training process. It's particularly useful when it's difficult to extract relevant features from data and when you have a high volume of data.

- **Reinforcement learning:** Is a machine learning method that is neither supervised learning nor unsupervised learning[13]. It trains software to make decisions that lead to optimal results. The method imitates the human process of learning through trial and error to achieve goals.

Table 2.1 below provides a comparison of these machine learning methods, highlighting their descriptions, applications, and challenges.

Table 2.1: Comparison of Machine Learning Methods

Methods	**Description**	**Applications**	**Challenges**
Supervised Learning	Uses labeled datasets to train algorithms for classification and regression tasks.	Classification, Regression	Requires labeled data, may overfit
Unsupervised Learning	Analyzes and groups unlabeled datasets to find hidden patterns.	Clustering, Association	More challenging to implement, not as widely used
Semi-supervised Learning	Hybrid approach that combines labeled and unlabeled datasets. Useful when feature extraction is difficult and for large datasets.	Various machine learning tasks	Requires careful handling of unlabeled data, may not perform well with limited labeled data
Reinforcement Learning	Trains software to make decisions through trial and error to achieve goals.	Game playing, Robotics	High computational complexity, requires careful tuning of reward functions

2.2.2 Deep Learning

Deep learning is one of the main Machine Learning (ML) technologies that uses artificial neural network algorithms designed to function similarly to the human brain learn from large amounts of data.
deep learning uses multiple layers of nonlinear processing units to extract and transform features. Higher layers learn more sophisticated characteristics that are deduced from the simpler features learned by the lower layers, which are closer to the data input. This architecture creates a hierarchical and powerful feature representation .
deep learning is appropriate for both large-scale data analysis and knowledge extraction from disparate sources as well as massive volumes of data[1].

2.3 Artificial Neural Networks

Neural networks or Artificial Neural Networks (ANNs) are a subset of machine learning and represent a type of deep learning technology, their structure mimics the way that biological neurons signal to each other. ANNs are composed of multiple nodes also known as artificial neuron containing an input layer, one or more hidden layers, and an output layer, each node, or artificial neuron, connects to another and is characterized by its weight, bias, and activation function[17] (as shown in Figure 2.2). These networks are used in image recognition and natural language processing (NLP).

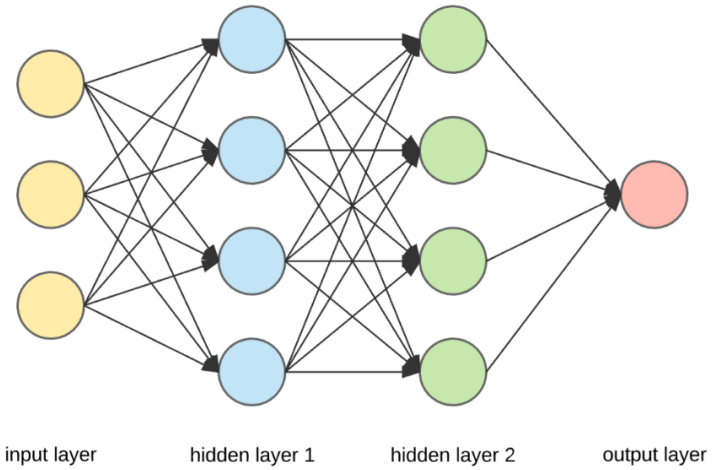


Figure 2.2: Artifitial Neural Network architecture

2.3.1 The neural network layers

The Neural Network has 3 type of layers, each of which will be explained below:

- **Input layer:** The input layer is the first layer of a neural network that receives the input data and sends it to the hidden layer(s) for further processing , each neuron corresponds to one feature of the input data[26][27].
- **Hidden layer:** The second type of layer in a neural network is the hidden layer, these are the layers between the input and output layers, composed of neurons, each of them is connected to every neuron in the adjacent layers. These layers consist of several interconnected nodes that perform computations on the data received from the input layer[26][27].
- **Output layer:** It is the last layer of a neural network , it represents the final prediction generated by the network, This layer may consist of one or more nodes , at each node, an activation function is applied to transform the weighted sum of the inputs from the previous layer into an output depending on the problem being solved[27].

2.3.2 Weight

Weight is a crucial component in Artificial Neural Networks (ANNs) , as it controls the strength of the connection between inputs and neurons. Each connection between neurons has an associated weight, representing scalar multiplications [18].

2.3.3 Bias

Biases are essential components of a perceptron serving as constants added to the weighted sum of inputs before applying an activation function. They provide an additional input of 1 into the next layer. Unlike the other inputs, biases are not influenced by the previous layer, but they

do have outgoing connections with their own weights. They help shift the activation function towards the positive or negative side.

2.3.4 Neurons

Neurons, also referred to as nodes, are the fundamental units of a neural network. They are typically represented as circles or dots in a diagram. These neurons are organized into layers and are responsible for performing computational tasks such as processing input and generating output according to activation functions[26]. Each neuron applies an activation function to the weighted sum of its inputs in order to process them, as expressed by the equation :

$$\text{output} = f(X) = f\left(\sum_{i=1}^{n} p_i x_i + z_{\text{bias}}\right) \tag{2.1}$$

Where f is the activation function used to activate or not the neuron, n is the number of input nodes, and $z_{\text{bias}} = W_0$ is the bias term used to adjust the output and $p_i x_i$ the weighted sum of the inputs. This process is illustrated in a simple perceptron model, as shown in Figure 2.3.

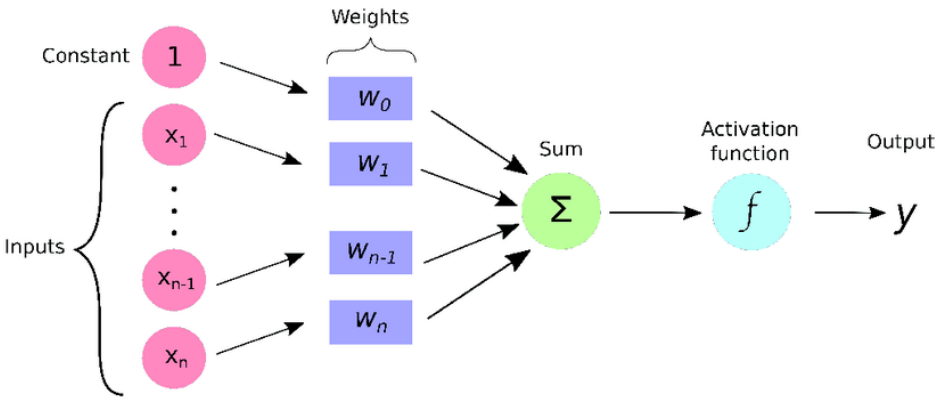


Figure 2.3: Simple perceptron.

2.3.5 Activation function

The activation function is a crucial component of neural networks as it introduces non-linear variations in the neural network. It defines whether a given node should be 'activated' based on the weighted sum of inputs[19]. Below are the most widely used functions:

– **Sigmoid function:** The sigmoid function also known as the logistic function is a mathematical expression that maps input values to an output range between 0 and 1 , it produces the curve that will have the shape "S", represented by the following equation:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

– **ReLU function:** It is a mathematical operation that replaces all the negative values with zero, while leaving the positive values unchanged, and it is represented by the following formula:

$$f(x) = \max(0, x) \tag{2.3}$$

– **Tanh function:** The hyperbolic tangent (tanh) function is quite similar to the sigmoid function, is a mathematical expression that converts the input values from -1 to 1, as given by the following formula:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

– **Softmax function:** Softmax is an activation function that transforms the input values into values between 0 and 1 , this allows them to be interpreted as probabilities. It is frequently used as the final layer of a model. Mathematically, Softmax is defined as follows:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{2.5}$$

Table 2.2: Summary of Activation Functions

Activation Function	**Mathematical Expression**	**Output Range**
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$[0, 1]$
ReLU	$f(x) = \max(0, x)$	$[0, \infty]$
Tanh	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$[-1, 1]$
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$	$[0, 1]$

2.3.6 Loss function

The loss function, also known as the error function, is a crucial aspect of neural networks. It compares the target and predicted output values to measure how well the neural network models the training data, therefore, the optimization of a model requires the reduction of its loss function[20].

The categorical cross-entropy function is a loss function commonly used in multiclass classification tasks. It is often employed to predict probabilities in neural networks[21]. The mathematical form would be :

$$L_{\text{CE}} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{2.6}$$

where y_i is the predicted value, and $\hat{y}_i$ is the result of the SoftMax function for the i^{th} class.

2.4 Convolutional Neural Network(CNN)

In 1998, LeCun et al. presented the LeNet architecture, the first Convolutional Neural Network (also known as CNN or ConvNet). It is a unique kind of neural network that was initially utilized in the field of computer vision and was mainly used for character recognition and Optical Character Recognition (OCR) tasks in documents[1]. The design was inspired by the human visual cortex, which is an animal brain's visual process[2].
CNN analyzes incoming data using a particular mathematical technique known as convolution, This operation is better than matrix multiplication because it is shift-invariant and computationally efficient, making it ideal for processing pixel data[25]. An example of convolutional neural network architecture for image classification is shown in Figure 2.4.

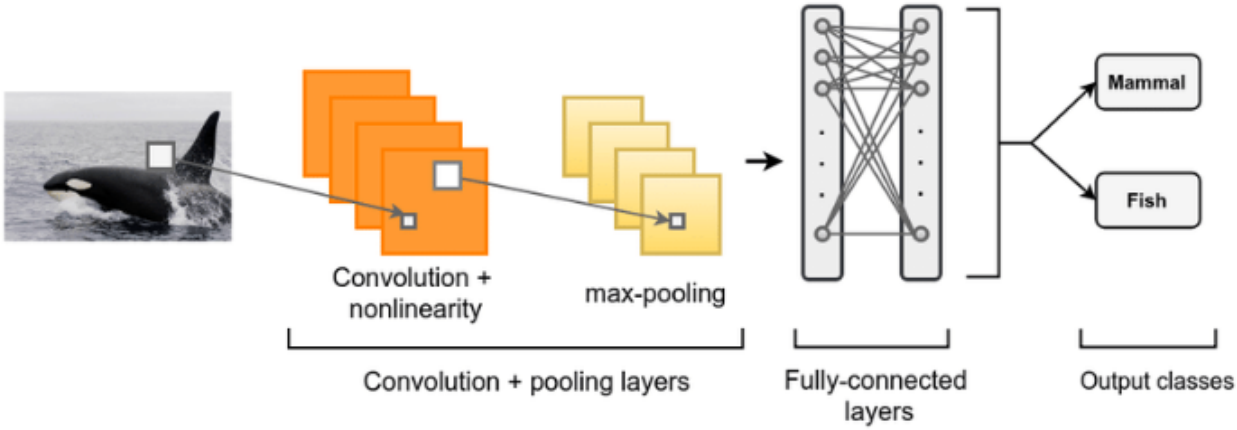


Figure 2.4: Sample convolutional neural network architecture for image classification

CNN has three unique layers: convolutional, pooling, and fully connected, each layer serves a specific purpose.

- **Convolutional Layer:**The convolution layer is the most important part in the CNN, responsible for performing the convolution operation which involves computing the dot product between a kernel matrix which contains the learnable parameters crucial for feature extraction, and a restricted portion of the input image known as the receptive field , this implies that when the image has three color channels (RGB) the kernel's size in terms of height and width will be spatially small while its depth will encompass all three channels[15].
  The convolutional layer transforms input data, typically an image, of size $N \times C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}$ to an output feature map of size $N \times C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}}$ via a convolution operation. The equation used to express this process is as follows:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k) \tag{2.7}$$

The symbol * represents the convolution operation, where *N* represents the batch size, *Cin* and *Cout* denote the number of input and output channels, respectively, and *Hin*, *Win*, *Hout*, and *Wout* represent the height and width dimensions of the input and output feature maps.

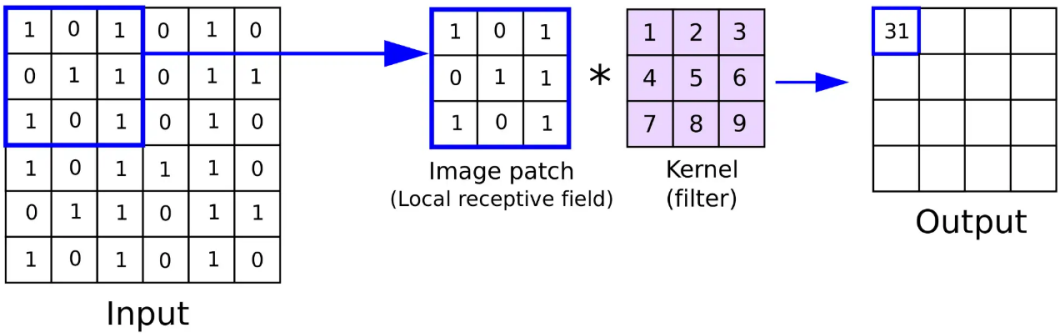


Figure 2.5: The convolution process

- **Pooling Layer:**Pooling layers are used to minimize the spatial dimensions of the image representation, and are typically inserted between two convolutional layers. These layers have two important hyperparameters: window size and stride, which determine the distance they move across the image. There are two main types of pooling: Max Pooling, which selects the highest value from every feature map window . the result would be a feature map with the most prominent features from the prior feature map. and Average Pooling, which computes the average of elements in every area of the feature map covered by the filter[16], as shown in Figure 2.6.

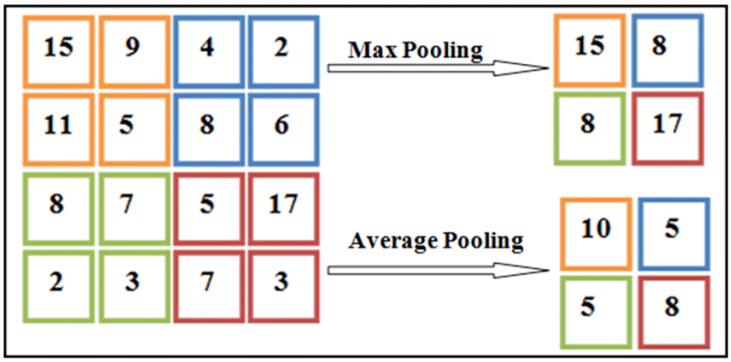


Figure 2.6: The process of different pooling layer

- **Fully Connected Layer:** The fully connected layers also referred to as dense layers, come after the convolutional and pooling layers In a Convolutional Neural Network. They bring together the learned representations from the convolutional layers, which have been selectively activated through the pooling layers, which is flattened and then fed into the fully connected layer to make predictions or classifications[23].

The output of a Fully Connected Layer is computed through a combination of an Affine function and a Non-Linear function. The Non-Linear function incorporates activations such as Sigmoid, TanH, and ReLU after the Affine function. Therefore, the computation can be expressed as:

$$y = \phi(Wx + b) \tag{2.8}$$

where ϕ denotes the activation function, W is the weight matrix, x is the input vector, b is the bias vector, and y is the output vector[24].

2.5 Recurrent Neural Networks

A recurrent neural network (RNN) is an artificial neural network that is designed to work with sequential or time series data[14]. while RNNs are very practical for sequential data processing tasks, they may encounter difficulties in learning long-term dependencies[22]. Therefore, researchers often use a variant of RNNs known as long short-term memory (LSTM) networks[25]. The architecture of recurrent neural networks is illustrated below in Figure 2.7.

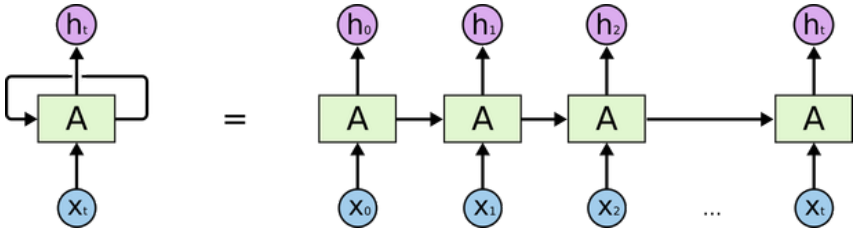


Figure 2.7: Architecture of Recurrent Neural Networks

2.6 Long Short-Term Memory networks

Long Short-Term Memory network (LSTM) was presented by Hochreiter and Schmidhuber in 1997[1]. It is a type of Recurrent Neural Network (RNN) that can handle and process sequential data. The LSTM network is structured with a series of LSTM cells, Each cell has input, output, and forget gates that control the flow of information into and out of the cell. The gates are utilized to selectively retain or discard information from previous time steps, enabling the network to maintain long-term dependencies on incoming data [3].

The LSTM unit illustrated in Figure 2.8 is the basic component of an LSTM architecture. It is a series of gates and cells that cooperate to produce a final result. An LSTM forward pass is expressed by the equations :

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c), \\
c_t &= f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t, \\
h_t &= o_t \otimes \tanh(c_t),
\end{aligned}
\tag{2.9}
$$

where σ denotes the sigmoid function, f_t is the Forget gate activation vector, i_t is the input gate activation vector, o_t is the output gate activation vector, $\tilde{c}_t$ is the cell input activation vector, c_t is the cell state, h_t is the output vector of the LSTM unit, and W and U are weights. The biases b and the symbol $\otimes$ represent the element-wise (Hadamard) product. The weights W, U, and biases b must be learned during the training process[22].

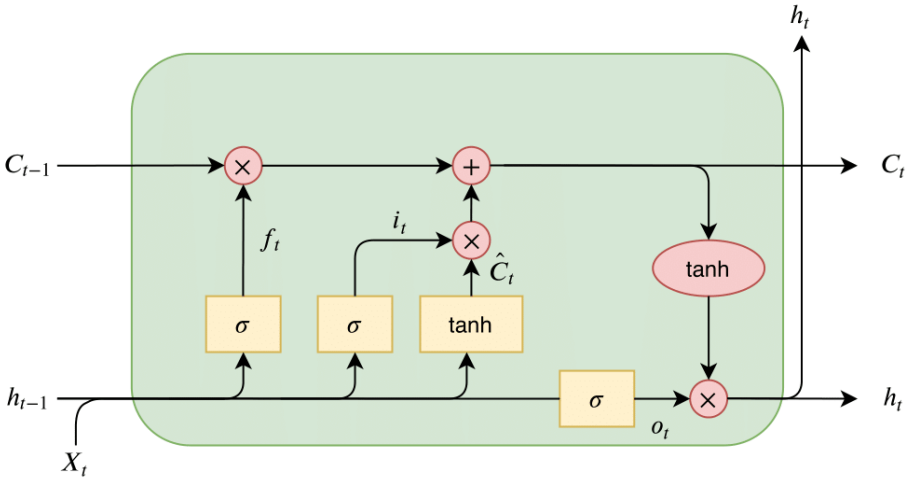


Figure 2.8: LSTM basic unit

2.7 Autoencoder

Autoencoders (AE) are a type of feedforward neural network where the input is the same as the output. They were developed by Geoffrey Hinton in the 1980s to address unsupervised learning problems. The input is compressed into a lower-dimensional code and then reconstructed into the output from this representation [5].
An autoencoder consists of three main components: an encoder, a bottleneck(code), and a decoder, as shown in figure2.9. The encoder compresses the input and produces the code; the decoder then reconstructs the input only using this code [6]. Autoencoders can be considered powerful tools for both dimensionality reduction and unsupervised learning tasks [4].

Encoder: The encoder is responsible for compressing the input data, such as images into a

latent space representation by encoding it in a reduced dimension. This compression is achieved through a sequence of layers which sequentially decrease the dimension of their outputs[4][7].

Bottleneck: It could be argued that the bottleneck of this architecture is corresponds to the layer at the end of the encoder layer, as it represents the compressed input that is then fed to the decoder layer and contains the most important features of the input data. It is worth noting that its dimension is the smallest among the layers of the network. [4][7].

Decoder: The decoder is used to restore the encoded image to its original dimensions. This is achieved by utilizing a stack of layers whose dimensions progressively increase until they match the input dimension. As a result, a lossy reconstruction of the original image is obtained from the latent space representation[4][7].

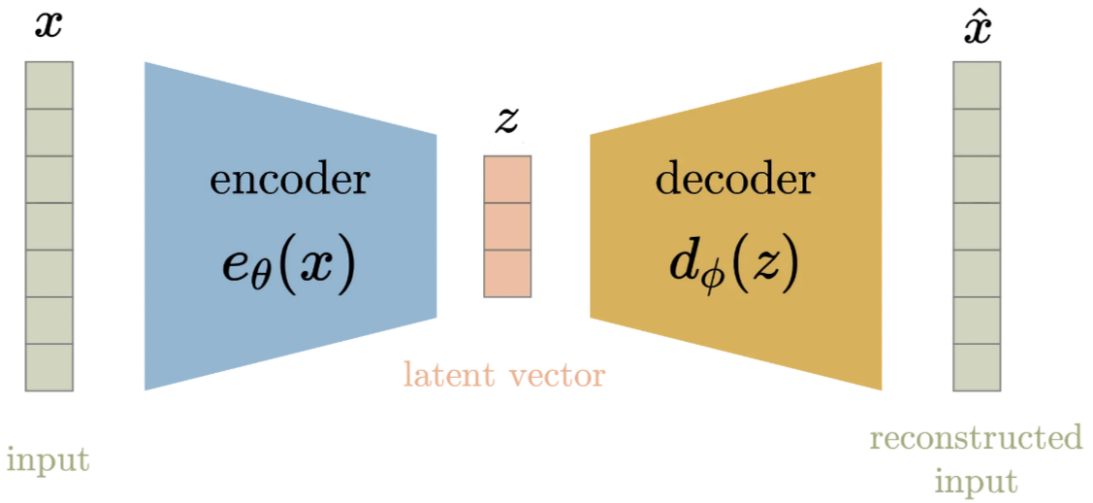


Figure 2.9: Autoencoder architecture

A loss function is used to quantify the difference between the input and the reconstructed data when the autoencoders are trained. One commonly used loss function is the squared Euclidean distance, which is as follows:

$$\text{loss (MSE)} = ||x - \hat{x}||^2 = ||x - d\phi(z)||^2 = ||x - d\phi(e^\theta(x))||^2 \tag{2.10}$$

Where z represents the latent representation (encoder output), x represents the input data, $\hat{x}$ represents the reconstructed data, ϕ and d are the encoder and decoder functions, respectively, θ denotes the parameters of the encoder function and e denotes the parameters of the decoder function [8].

2.8 Evaluation metrics of models

In the field of machine learning, where models are assigned the task of making predictions or classifications, evaluating their performance stands as a critical endeavor. Utilizing evaluation metrics provides a means to gain a clear understanding of a model's efficacy and facilitates comparisons and improvements. To determine the accuracy and dependability of these models, we depend on a set of metrics that contrast the model's predictions with the ground truth or actual labels. At the core of these evaluation metrics lie four fundamental concepts that are commonly employed in assessing the performance of classification models: True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). These foundational terms are delineated below:

- **True Positive (TP):** refers to a correct prediction where an instance belonging to the positive class is classified accurately[50].
- **True Negative (TN):** indicates a correct prediction where an instance belonging to the negative class is classified accurately[50].
- **False Positive (FP):** signifies an incorrect prediction where an instance from the negative class is wrongly classified as belonging to the positive class[50].
- **False Negative (FN):**represents an incorrect prediction where an instance from the positive class is classified wrongly as belonging to the negative class[50].

After outlining these fundamental concepts, we delve into various evaluation metrics, which are outlined below:

– **Accuracy:** Is the percentage of correctly classified instances out of all instances in the dataset, serving as a fundamental metric for classification tasks and measuring the overall performance of the model[51][53]]. It is calculated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

– **Precision:** Precision measures the proportion of true positives out of all cases classified as positive, including both true positives and false positives. It focuses on the accuracy of positive predictions[51][52][53]. The mathematical representation of this metric is:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.12}$$

– **Recall (Sensitivity):** Recall measures the ratio of true positives among all actual positive instances (true positives + false negatives). It emphasizes the model's ability to identify all relevant instances[51][52][53]. The formula for this metric is:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.13}$$

- **F1-Score:** The F1 score, serving as a balance between precision and recall, integrates both metrics into a single measure. This makes it particularly useful when dealing with imbalanced datasets. Representing the harmonic mean of precision and recall, it offers a comprehensive evaluation of model performance[51][52][53]. The expression for this metric is:

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.14}$$

- **Area Under the Receiver Operating Characteristic Curve (AU-ROC):** AU-ROC summarizes the ROC curve, representing the trade-off between true positive rate and false positive rate. The AU-ROC value ranges from 0 to 1, where a higher value indicates better model performance[51].

- **The Area Under the Precision-Recall Curve (AUPRC):** is a metric used to evaluate binary classification models, especially on imbalanced datasets. It measures the area under the Precision-Recall curve, showing the balance between precision and recall at different thresholds. A higher AUPRC value (0 to 1) indicates better model performance, with 1 being perfect. AUPRC focuses on positive class performance and is preferred for imbalanced datasets compared to AUROC[58][59].

- **Log Loss:** Log Loss, also known as logistic loss or logarithmic loss, is a performance metric for classification models that measures the difference between predicted probabilities and actual labels . It provides insight into the model's calibration and accuracy, indicating how close a prediction probability aligns with the actual true value[55][56][57]. The formula for Log Loss is given by:

$$\text{Log Loss} = -\frac{1}{n} \times \sum_{i=1}^{n} [y_{\text{true}}^{(i)} \times \log(y_{\text{pred}}^{(i)}) + (1 - y_{\text{true}}^{(i)}) \times \log(1 - y_{\text{pred}}^{(i)})] \tag{2.15}$$

- **Confusion matrix** A confusion matrix, also known as an error matrix, is a table used to evaluate the performance of an algorithm by comparing predicted labels with actual labels. In a binary classification, the matrix consists of four main components: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).
  This matrix (Figure 2.10) provides a clear visualization of an algorithm's performance by showing the accuracy of predictions across different categories and helps in calculating various performance metrics such as accuracy, precision, recall, and F1 score.

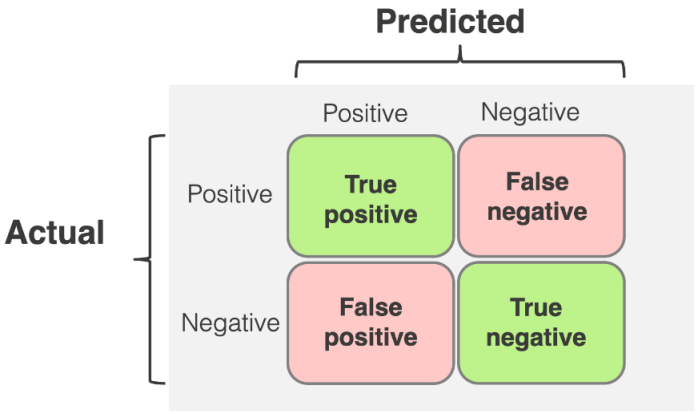


Figure 2.10: Confusion matrix Table structure

2.9 Related Works

Word boundary prediction is a crucial task in continuous sign language processing, aiming to identify the boundaries between words in a continuous stream of sign language gestures. This task presents challenges similar to those encountered in speech processing, as there are often no clear visual cues indicating where one sign ends and another begins. Accurate word boundary prediction is essential for various natural language processing applications, including sign language recognition, translation, and synthesis.

Several approaches have been explored for word boundary prediction in sign language recognition. Traditional statistical models such as hidden Markov models (HMMs) and conditional random fields (CRFs) have been utilized to segment continuous sign language streams into individual signs [33]. These models leverage probabilistic frameworks to identify transitions between signs based on observed features.

In recent years, neural network-based approaches have gained prominence in word boundary prediction for continuous sign language recognition [34]. Deep learning models, including recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer architectures, have shown promise in capturing the temporal dependencies and complex patterns present in continuous sign language data.

Zhou et al. and Wei et al. worked on developing models capable of handling the variability and complexity of sign language, with a specific focus on reinforcement learning techniques for semantic boundary detection [37, 38]. Khanh et al. focused on the effective segmentation of word gestures in sign language videos to enhance CSLR accuracy [36]. Yu et al. introduced SkeFormerSL, a model incorporating skeletal input data to improve sign language segmentation accuracy [41]. These works demonstrate advances in deep learning techniques to address challenges in continuous sign language processing, particularly with a focus on word boundary prediction and segmentation accuracy.

Using advanced machine learning techniques, researchers aim to enhance the efficiency and accuracy of word boundary prediction in continuous sign language recognition systems. Future

research directions may focus on exploring advanced neural network architectures, incorporating multimodal information, and optimizing model performance for real-time sign language recognition tasks.

2.10 Conclusion

In this chapter, we provided a thorough overview of deep learning concepts, including artificial neural networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), autoencoders, and Long Short-Term Memory (LSTM) networks. We critically reviewed the current state of research, highlighting the latest advancements in machine learning and deep learning technologies. In the upcoming chapter, we will discuss the steps involved in implementing our model for word boundary prediction in continuous sign language.

Chapter 3

Design of the BPredictor4CSL system

3.1 Introduction

In this chapter, we explore the intricate design and architecture of the BPredictor4CSL system, a cutting-edge solution for predicting sign language segment boundaries in continuous signing videos. We commence by formally defining the problem formulation, emphasizing the system's objective of accurately labeling frames within a video sequence to delineate the transitions between different signing gestures. This is followed by an overview of the BPredictor4CSL system's architecture, which outlines the four distinct phases involved in the process: data preparation, preprocessing, model training and evaluation, and model prediction and evaluation.

The chapter then provides a detailed exposition of the preprocessing phase, which plays a crucial role in extracting relevant features and enhancing the quality of the raw video data. We meticulously describe the steps involved, including video loading and frame extraction, facial and hand landmark detection, feature extraction, dataset labeling, and resource release and cleaning procedures. These steps are essential for transforming the raw video data into a structured and informative dataset, suitable for subsequent analysis and model training.

Through an exploration of the BPredictor4CSL system and its preprocessing phase, this chapter aims to provide valuable insights into the complexities of sign language segment boundary prediction. By elucidating the system's architecture and preprocessing procedures, we strive to advance our understanding of sign language processing technology and its potential applications in real-world scenarios.

The chapter then provides a detailed exposition of the preprocessing phase, which plays a crucial role in extracting relevant features and enhancing the quality of the raw video data. We meticulously describe the steps involved, including video loading and frame extraction, facial and hand landmark detection, feature extraction, dataset labeling, and resource release and cleaning procedures. These steps are essential for transforming the raw video data into a structured and informative dataset, suitable for subsequent analysis and model training.

3.2 Problem Formulation

In the context of continuous signing in sign language, represented as a sequence of video frames $F = (f_1, f_2, ..., f_n)$, the primary objective of the *BPredictor4CSL* (Boundary Predictor for Continuous Sign Language) system is to predict a corresponding vector of labels $y = (y_1, y_2, ..., y_N)$. These labels serve to delineate between different segments of signing within the continuous sequence. Specifically, the label values of 1 and 0 are employed, where:

- A value of 1 indicates that the signer is still actively engaged in signing.
- A value of 0 signifies the presence of boundaries between consecutive sign segments, thereby demarcating the transitions between different signs.

The challenge lies in accurately predicting these boundary labels within the continuous signing sequence, taking into account the dynamic nature of sign language expressions, variations

in signing speeds, hand shapes, facial expressions, and body movements. The *BPredictor4CSL* system endeavors to leverage advanced computer vision techniques and deep learning methodologies to effectively analyze and interpret the visual cues present in the video frames, enabling precise segmentation of sign language gestures. By providing reliable predictions of sign segment boundaries, the system aims to enhance the efficiency and accuracy of automated sign language recognition, thereby facilitating seamless communication for individuals who rely on sign language as their primary mode of expression.

As illustrated in Figure 3.1, the BPredictor4CSL system processes each frame in the sequence and predicts the boundary labels y=(y_1, y_2,...,y_n), where each label delineates active signing or boundaries between sign segments.

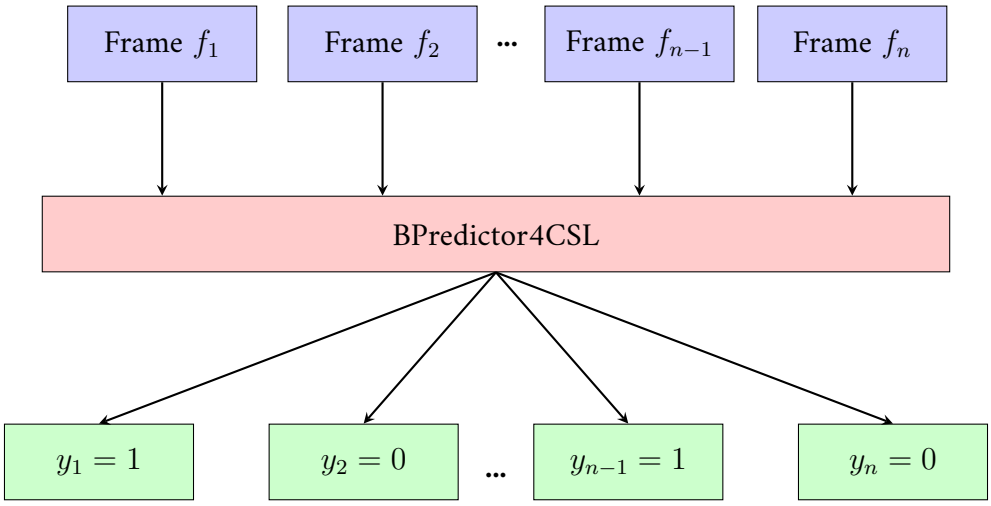


Figure 3.1: BPredictor4CSL: Predicting Boundary Labels for Continuous Sign Language

3.3 Overview of the BPredictor4CSL system

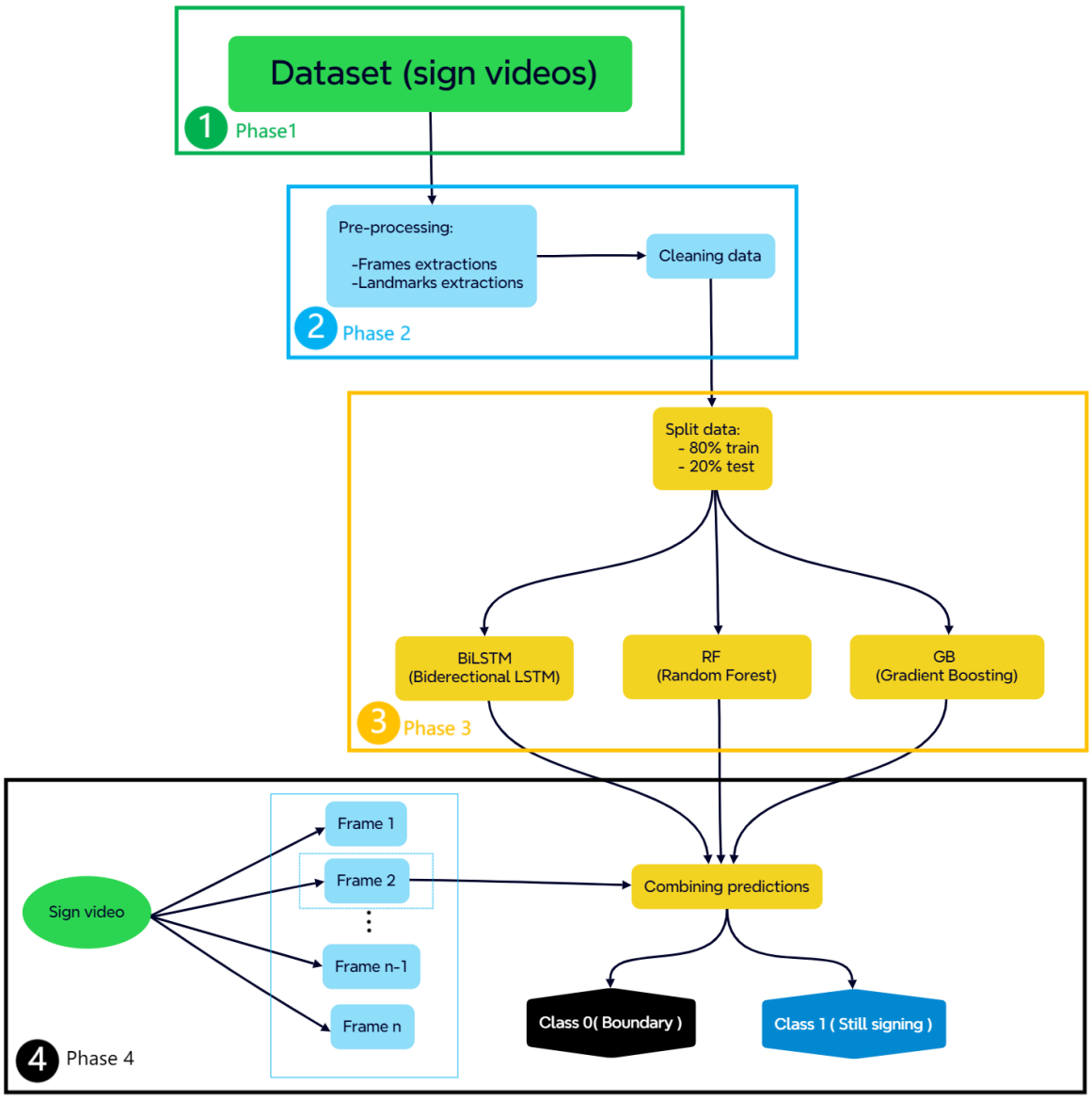


Figure 3.2: General diagram of the BPredictor4CSL

In our BPredictor4CSL system, we implement four essential phases. Phase 1 involves defining the folder paths for CSV files containing annotations and the folder containing video files. These annotations provide crucial information, such as video ID, tier ID, and the number of frames. Additionally, we establish the necessary face and hand landmarks and their corresponding indices for feature extraction. In Phase 2, we undertake preprocessing tasks, including landmark extraction and CSV generation. Functions are defined to extract relevant landmarks from video frames using the MediaPipe Hands and FaceMesh models. These landmarks encompass facial features (such as eyebrows, eyes, and mouth) and hand landmarks. We then iterate through each video and its associated CSV file containing annotations. For each row in the CSV file, representing a specific segment of the video, we filter the frames based on the tier ID, process each

frame to extract landmarks and labels, and subsequently write this data to the CSV file. Phase 3 focuses on training a Hybrid learning model employing a Sequential architecture with Bidirectional LSTM (BI-LSTM) layers , Random Forest (RF) and Gradient Boosting (GB) . The data undergoes preprocessing,. Finally, in Phase 4, the trained neural network model and ensemble models are employed to predict labels for new data. The predictions are assessed through confusion matrices and classification reports, and similarity percentages between actual and predicted labels are calculated for further analysis.

3.4 Pre-Processing

In this section, we focus on the preprocessing phase, a critical stage in our data analysis pipeline. Here, our primary objective is to prepare the raw video data for thorough analysis and effective model training. We begin by transforming raw annotation data from ELAN files into a format suitable for analysis and modeling. This step is particularly essential when dealing with datasets like the DGS Corpus, which provides annotations in ELAN format. Converting the data to CSV format is essential for streamlining workflow and improving data handling. Following this, we delve into a detailed series of steps aimed at extracting relevant features and enhancing the overall quality of the data

1. **Conversion to CSV Format:**

   The first step involves converting ELAN annotation files into CSV format. This conversion simplifies data handling and enables easier manipulation and analysis of the annotation data. Each annotation segment is extracted, capturing relevant details such as tier ID, annotation ID, start time, end time, and annotation value. This structured representation ensures organized storage and facilitates subsequent processing.

2. **Calculation of Frame Counts:**

   In addition to conversion, the phase includes the calculation of frame counts for each annotation segment. By leveraging assumptions about the frame rate (e.g., 30 frames per second), the duration of each segment is converted from milliseconds to seconds. Multiplying this duration by the frame rate yields the number of frames corresponding to each segment. This temporal context enriches the dataset, providing valuable information in terms of video frames. The calculation of frames aims to facilitate the data labeling step, ensuring that each annotation segment is associated with the appropriate temporal extent in terms of video frames.

3. Video Loading and Frame Extraction

The initial step in the video processing pipeline is to load the video data from the source files. This is commonly accomplished using libraries such as OpenCV in Python, which provide efficient methods for handling video files. OpenCV allows for seamless video loading, decoding, and manipulation.

Once the video is loaded into memory, the next step is to extract individual frames from each video file. Each frame represents a single image snapshot captured at a specific point in time during the video. These frames serve as the basic units of analysis and are used for subsequent processing steps.

Frame extraction involves iterating through the video file and capturing each frame sequentially. OpenCV provides functions to read frames from a video file and store them as image arrays. Depending on the requirements of the application, frames can be extracted at regular intervals or based on specific criteria, such as keyframe extraction for video summarization.

The extracted frames are typically stored as image files or in memory as arrays for further analysis. These frames serve as the foundation for various computer vision tasks, including object detection, tracking, and recognition. Additionally, they enable the analysis of temporal dynamics and motion patterns within the video content.

By leveraging libraries like OpenCV for video loading and frame extraction, the video data becomes accessible for further processing, enabling a wide range of applications in computer vision and video analysis.

4. Facial and Hand Landmark Detection

Following frame extraction, the subsequent step is the detection of facial and hand landmarks within each frame. This process involves utilizing deep learning models, such as MediaPipe, which offers pre-trained models specifically designed for landmark detection tasks. These models are highly efficient and capable of accurately detecting key points on the face, such as the eyes, nose, and mouth, as well as on the hands, including fingertips and palm, in real-time.

The detected landmarks provide crucial information about the spatial configuration of facial and hand gestures. For instance, facial landmarks help in identifying expressions, eye movements, and head orientation, while hand landmarks offer insights into hand positions, finger movements, and gestures. This spatial information is fundamental for sign language recognition, as it enables the system to interpret and understand the gestures and expressions made by the signer.

By leveraging deep learning models for facial and hand landmark detection, the system

can efficiently and accurately analyze each frame of the video, providing a detailed understanding of the signer's movements and expressions.

In this implementation, the following landmarks are extracted:

Features	Landmarks	Number of Landmarks
Face	Left Eyebrow	5
	Right Eyebrow	5
	Left Eye	3
	Right Eye	3
	Mouth	6
Hands	Hand 1	21
	Hand 2	21

Table 3.1: Number of Landmarks for Facial and Hand Landmarks

These specific numbers of facial landmarks ensure that critical regions such as the eyebrows, eyes, and mouth are accurately tracked, which are vital for understanding facial expressions and lip movements. For the hands, the landmarks cover all key points from the wrist to the fingertips, providing comprehensive data on hand positions and finger articulations.

Using these predefined numbers of landmarks, the system can focus on the most informative points, optimizing the analysis and recognition process for sign language interpretation.

5. Feature Extraction

After detecting landmarks for the hands and face within each video frame, the next critical step is the extraction of characteristics. This process involves several stages to derive meaningful insights from the detected landmarks.

Firstly, the coordinates (x, y, z) of selected landmarks, such as those representing the eyebrows, eyes, and mouth for the face, and similar key points for the hands, are extracted. These coordinates provide precise spatial information about the position and orientation of these landmarks within the frame.

Once extracted, these coordinates undergo filtering and organization to ensure that they are accurately represented and ready for further analysis. Filtering involves removing any noise or outliers that may have been detected during the landmark detection process. Additionally, the coordinates are organized in a structured manner to facilitate subsequent feature extraction.

Following filtering and organization, the extracted coordinates are transformed into a set of features. These features capture important characteristics of hand and facial movements observed in the video frames. For example, features derived from hand landmarks

might include measurements of finger flexion, hand shape, or hand movement speed and direction. Similarly, features from facial landmarks might include measurements of facial expressions, head movement, or eye gaze direction.

By representing these features, the extracted data becomes more meaningful and interpretable. These features provide valuable insights into the dynamics of hand and facial movements throughout the video sequence, laying the groundwork for further analysis and interpretation, as shown in Figure 3.3.

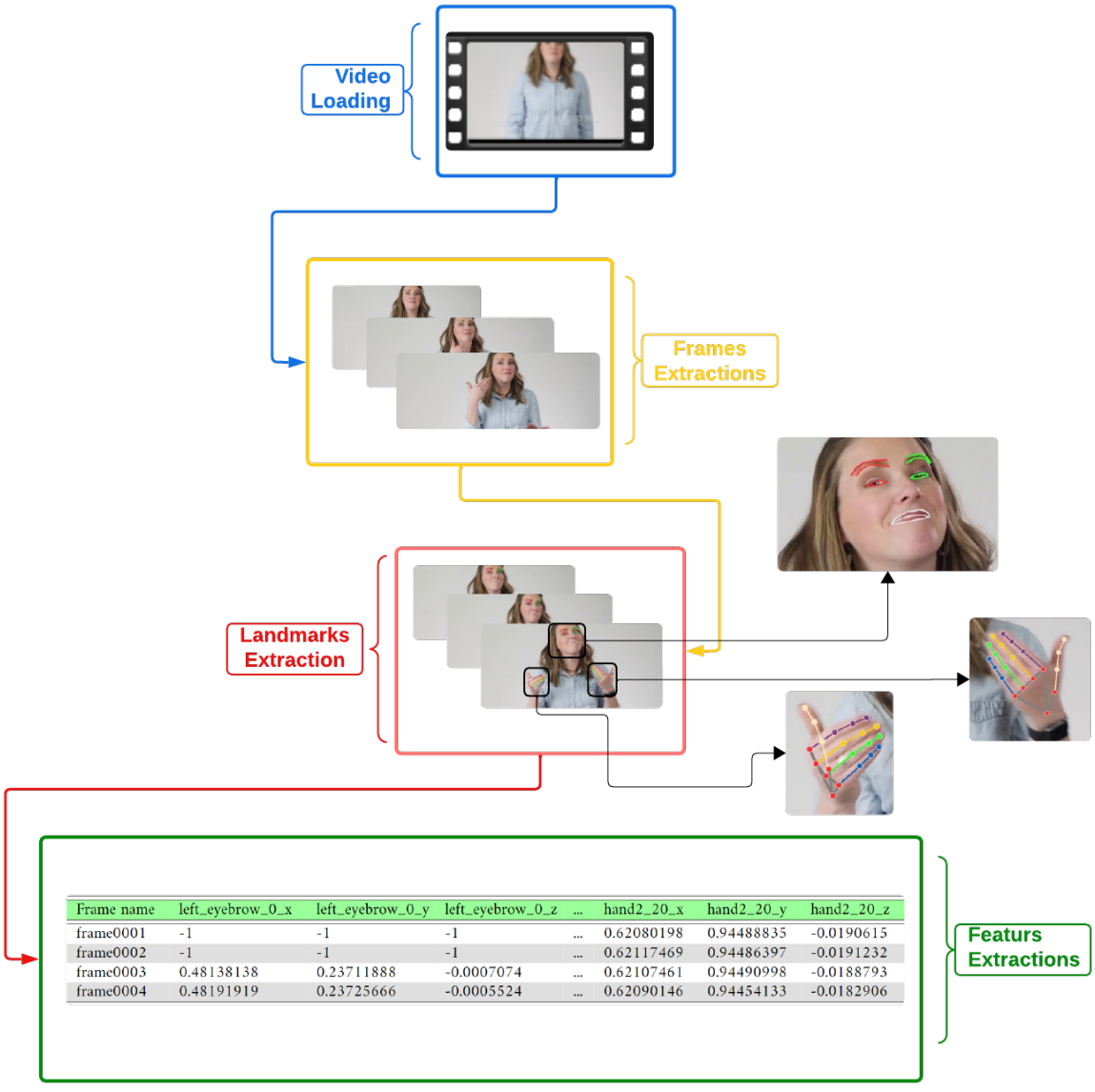


Figure 3.3: Pipeline for Feature Extraction from Video Frames

6. Labeling the dataset

Frames are labeled based on their position within signing segments: frames occurring within signing gestures are labeled as "still signing," while frames between segments are designated as "boundary frames." This labeling scheme enables for accurate classification of different segments in the signing videos (as shown in Figure 3.4). As frames are processed, the script writes the extracted landmarks and labels to a CSV file, forming the dataset for subsequent analysis and model training. Each row in the CSV contains the landmarks' coordinates and their corresponding labels, facilitating further exploration of the signing gestures. Special handling is implemented for boundary frames between signing segments, ensuring that an appropriate number of frames are extracted to maintain dataset integrity.

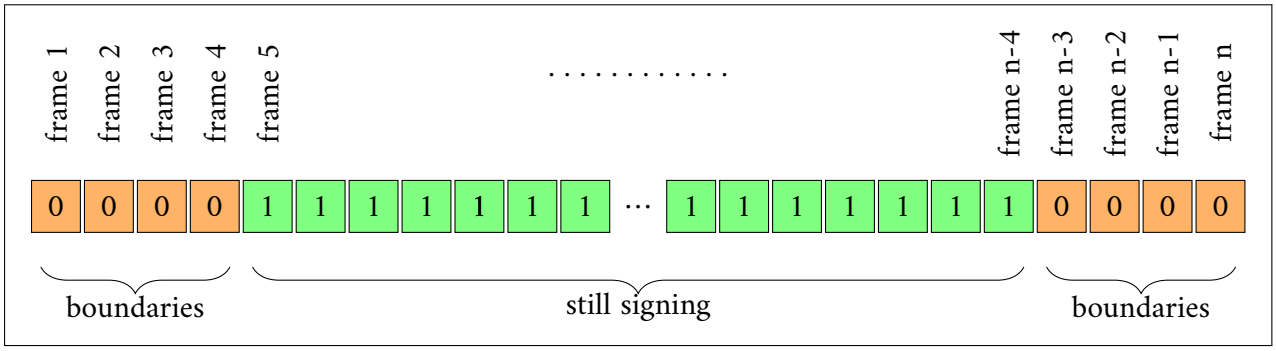


Figure 3.4: Sample Label Vector

7. Resource Release

After processing all frames in a video, resources such as video capture objects and MediaPipe models are released to optimize memory usage. This ensures efficient processing and prevents memory leaks.

8. Cleaning

After extracting landmarks and features from the video frames, the next step in the pre-processing phase involves cleaning the data to enhance its quality. This cleaning process includes several steps:

(a) **Detection of Non-Detected Landmarks**: The data is analyzed to identify frames where certain landmarks are not detected. This allows us to quantify the number of non-detected landmarks and assess their impact on the dataset.

(b) **Identification of Missing Value**s: The presence of missing values in the dataset is determined, indicating frames where essential information is absent.

(c) **Filling Missing and NaN Values**: Missing values and NaN (Not a Number) entries are filled to ensure consistency in the dataset. This step involves replacing missing values with appropriate placeholders or interpolated values.

By performing these cleaning steps, we ensure that the data is free from noise and inconsistencies, thereby optimizing it for subsequent analysis and modeling tasks.

These steps are illustrated in Table 3.2, which shows a sample of the dataset before cleaning, and Table 3.3, which presents the dataset after cleaning.

Table 3.2: A sample of created dataset before cleaning

Frame name	left_eyebrow_0_x	left_eyebrow_0_y	left_eyebrow_0_z	...	hand2_20_x	hand2_20_y	hand2_20_z
frame0001				...	0.62080198	0.94488835	-0.0190615
frame0002				...	0.62117469	0.94486397	-0.0191232
frame0003	0.48138138	0.23711888	-0.0007074	...	0.62107461	0.94490998	-0.0188793
frame0004	0.48191919	0.23725666	-0.0005524	...	0.62090146	0.94454133	-0.0182906

Table 3.3: A sample of created dataset after cleaning

Frame name	left_eyebrow_0_x	left_eyebrow_0_y	left_eyebrow_0_z	...	hand2_20_x	hand2_20_y	hand2_20_z
frame0001	-1	-1	-1	...	0.62080198	0.94488835	-0.0190615
frame0002	-1	-1	-1	...	0.62117469	0.94486397	-0.0191232
frame0003	0.48138138	0.23711888	-0.0007074	...	0.62107461	0.94490998	-0.0188793
frame0004	0.48191919	0.23725666	-0.0005524	...	0.62090146	0.94454133	-0.0182906

For a visual representation of the detailed pre-processing pipeline for video data, including all these critical steps, refer to Figure 3.5.

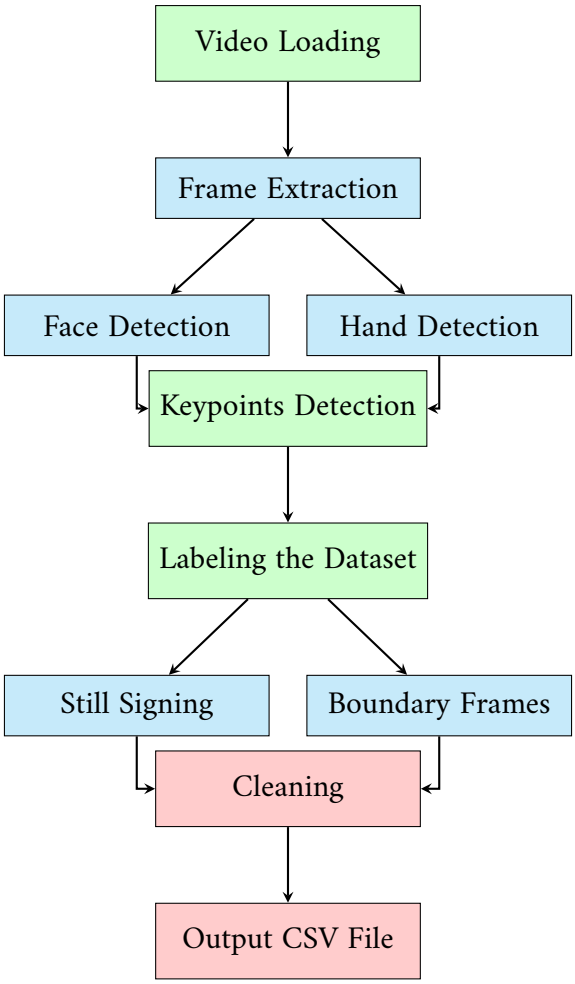


Figure 3.5: Detailed pre-processing pipeline for video data

3.5 Proposed Model

In this section, we present the detailed architecture and methodology of the BPredictor4CSL system's proposed model for detecting word boundaries in continuous sign language videos. This system leverages a combination of deep learning and traditional machine learning techniques to achieve high accuracy in segmenting sign language gestures. The hybrid approach combines the temporal modeling capabilities of Long Short-Term Memory (LSTM) networks and the robustness of ensemble classifiers. The following subsections describe the model architecture, training process, and evaluation methods in detail.

3.6 Model Architecture

The BPredictor4CSL system is composed of three main components: a deep learning model using Bidirectional LSTM (BiLSTM) layers and an ensemble of machine learning classifiers using Random Forest (RF) and the Gradient Boosting (GB). Each component is designed to address specific challenges associated with continuous sign language segmentation.

3.6.1 Bidirectional LSTM (BiLSTM)

The deep learning component of the BPredictor4CSL system is built using a Sequential model architecture with BiLSTM layers. This component captures the temporal dependencies and complex patterns inherent in sign language sequences (as shown in Figure 3.6).

- **Input Layer**: The input to the model is a sequence of frames, each represented by a feature vector derived from facial and hand landmarks. These features are extracted using advanced computer vision techniques to capture the nuances of sign language gestures.
- **Bidirectional LSTM Layers**: The architecture includes two BiLSTM layers. The first BiLSTM layer consists of 128 units, and the second has 64 units. The bidirectional nature of these layers allows the model to consider both past and future context, which is crucial for accurately identifying boundaries in a sequence. BiLSTM layers process the input sequence in both forward and backward directions, ensuring comprehensive temporal understanding.
- **Dropout Layers**: To avoid overfitting, dropout layers with a dropout rate of 0.5 are added after each BiLSTM layer. Dropout randomly deactivates neurons during training, promoting model generalization and robustness.
- **Dense Layer**: Following the BiLSTM layers, a dense layer with 64 units and ReLU activation is included. This layer incorporates L2 regularization to further enhance the model's generalization capabilities. The dense layer aggregates the temporal features captured by the BiLSTM layers, transforming them into a format suitable for classification.
- **Batch Normalization**: Batch normalization is applied after the dense layer to stabilize and accelerate the training process. This technique normalizes the output of the dense layer, reducing internal covariate shift and allowing for higher learning rates.
- **Output Layer**: The final layer is an output layer with a single neuron and a sigmoid activation function. This layer produces a probability score for each frame, indicating the likelihood of a boundary being present. A score close to 1 suggests a boundary, while a score close to 0 indicates continuous signing.

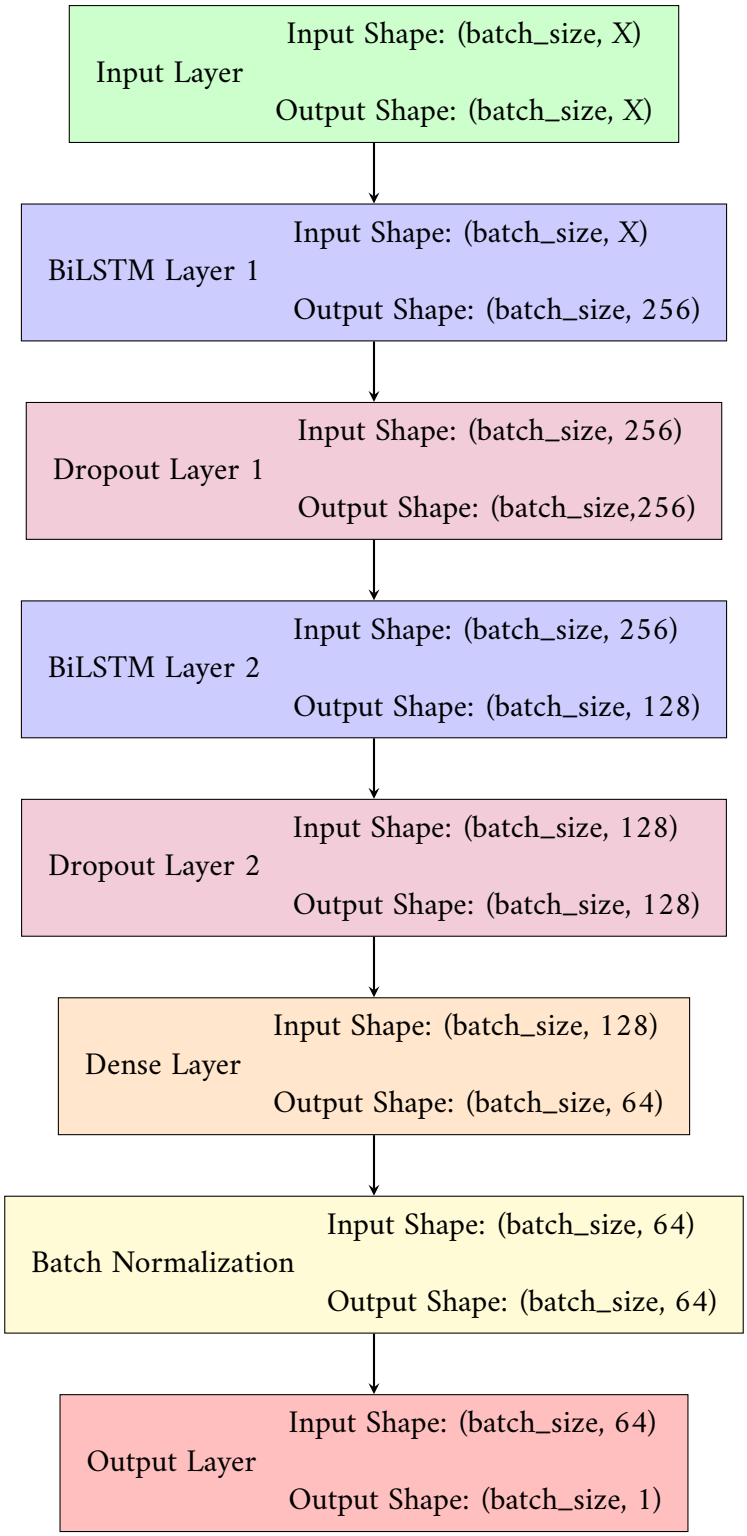


Figure 3.6: Detailed structure of the BiLSTM model, including input shape and output shape of each layer.

3.6.2 Random Forest Classifier Training

1. **Creating Lagged Features**: Each frame's features are shifted to create lagged features. This process captures the temporal dependencies in the sign language sequences, allowing the classifier to learn from historical context(as shown in Figure 3.7).

2. **Initialization**: The Random Forest classifier is initialized with a specified number of estimators, in this case, 100. Each estimator represents an individual decision tree within the forest.

3. **Training**: The classifier is trained on the prepared dataset using the ensemble learning technique. Each decision tree in the forest is trained on a random subset of the data and features, ensuring diversity among the trees. This randomness helps to capture different aspects of the data and mitigate overfitting.

4. **Feature Importance**: During training, the classifier computes feature importance scores, indicating the contribution of each feature to the classification process. These scores offer insights into which features are most influential for boundary detection, aiding in model interpretation.

5. **Evaluation**: After training, the performance of the Random Forest classifier is evaluated using a separate validation dataset or through cross-validation. Metrics such as accuracy, precision, recall, and F1-score are computed to assess the classifier's performance on unseen data.

3.6.3 Gradient Boosting Classifier Training

1. **Data Preparation**: Similar to the Random Forest classifier, the dataset is prepared by excluding non-feature columns and retaining relevant features for training. Additionally, lagged features are included in the dataset to provide the temporal context necessary for boundary detection(as shown in Figure 3.7).

2. **Initialization**: The Gradient Boosting classifier is initialized with a specified number of estimators, also set to 100 in this scenario. Each estimator represents a weak learner that sequentially improves the model's performance.

3. **Training**: The training process involves iteratively adding decision trees to the ensemble, with each tree focusing on the residual errors made by previous estimators. This sequential learning approach gradually improves the model's predictive capabilities by minimizing errors.

4. **Performance Improvement**: The Gradient Boosting algorithm continuously refines the model's predictions by emphasizing instances where previous estimators have performed poorly. This adaptive learning process enables the classifier to capture complex patterns and interactions within the data effectively.

5. **Evaluation**: Similar to the Random Forest classifier, the performance of the Gradient Boosting classifier is evaluated using appropriate metrics. The classifier's performance on unseen data is assessed to ensure its generalization capabilities.

The training of the ensemble classifiers involves iterative processes that leverage ensemble learning techniques to improve predictive accuracy. By combining multiple decision trees (in the case of Random Forest) or weak learners (in the case of Gradient Boosting), the ensemble classifiers capture diverse patterns within the data, enhanced by the inclusion of lagged features. This approach significantly improves the overall robustness and accuracy of the BPredictor4CSL system in detecting word boundaries in continuous sign language videos.

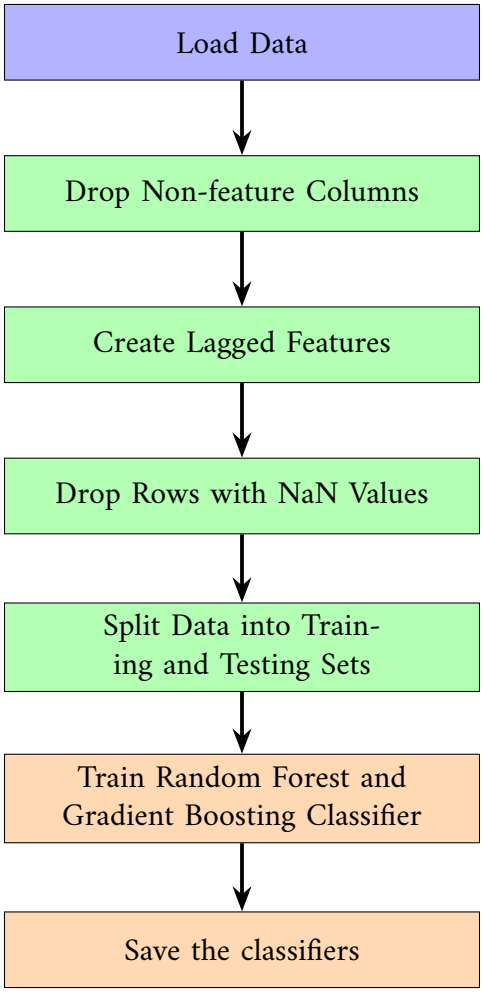


Figure 3.7: Data preparation for Random Forest and Gradient Boosting Classifiers training.

3.6.4 Combining Predictions

The final prediction is obtained by combining the outputs of the BiLSTM model and the ensemble classifiers. This hybrid approach ensures robustness and accuracy in predicting word boundaries, as illustrated in Figure 3.8.

- **Voting** :In this approach, the predictions from the BiLSTM model, Random Forest, and Gradient Boosting classifiers are combined using a voting mechanism. Each model casts a single vote for the predicted class, and the final prediction is determined by the majority vote. For instance, if two out of the three models predict a particular class, that class will be the final prediction. This method leverages the strengths of each individual model,

improving overall performance by balancing out the weaknesses of any single model. By combining the diverse perspectives of multiple classifiers, hard voting can enhance the robustness and accuracy of the predictions.

- **Thresholding**: The combined probabilities from the BiLSTM model, Random Forest, and Gradient Boosting classifiers are converted into binary predictions using a thresholding approach. Initially, the individual predictions are averaged to form a single probability for each frame. This combined probability is then compared to a predefined threshold, typically set to the average of the combined probabilities across all frames. If the combined probability for a specific frame is less than or equal than this average threshold, it is classified as a boundary (label 0). Conversely, if the combined probability greater to the threshold, it is classified as continuous signing (label 1). This thresholding step is crucial for finalizing the segmentation of the sign language video, ensuring each frame is accurately identified as either a boundary or part of continuous signing.

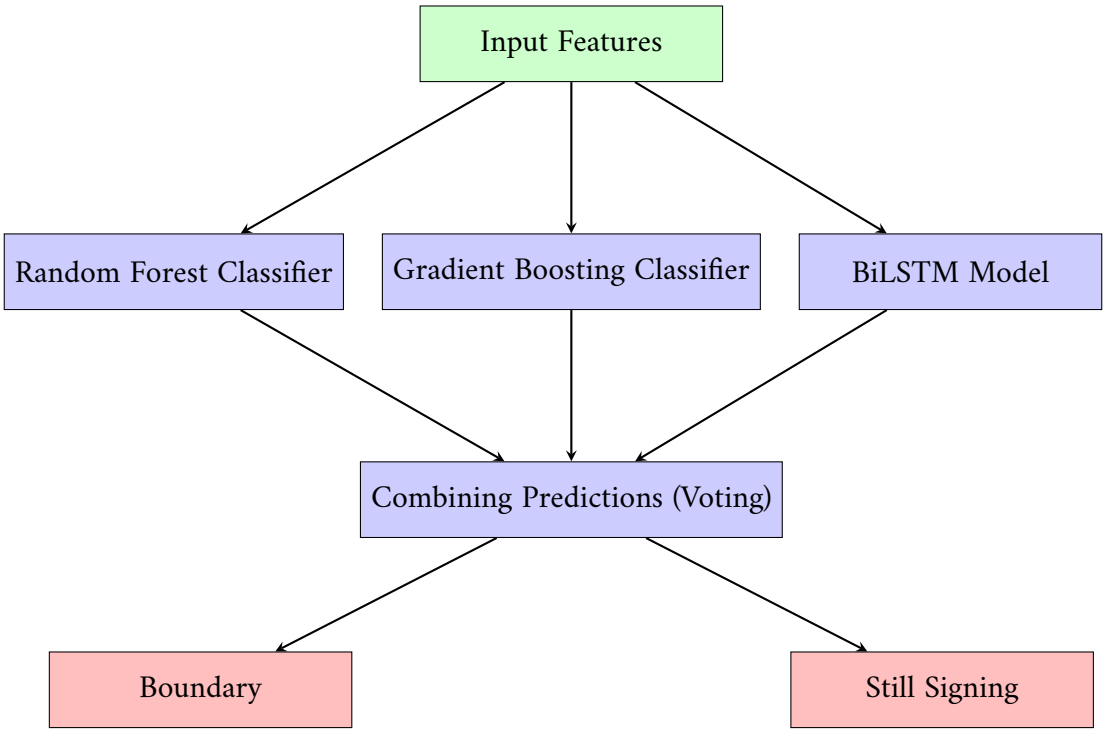


Figure 3.8: Ensemble Classifier Workflow: Combining Predictions from Random Forest, Gradient Boosting, and BiLSTM Models.

3.7 Conclusion

In conclusion, this chapter has comprehensively outlined the design and architecture of the BPredictor4CSL system, a pioneering solution for predicting segment boundaries in continuous sign language videos. By formally defining the problem formulation and providing an overview of the system's architecture, we have established a solid foundation for understanding the intricate workings of this innovative system.

Furthermore, the chapter has delved deep into the intricate details of the preprocessing phase, a critical stage in the overall process. Through meticulous descriptions of the various steps involved, including video loading, landmark detection, feature extraction, dataset labeling, and data cleaning, we have highlighted the importance of transforming raw video data into a structured and informative dataset, suitable for model training and analysis.

The insights and techniques presented in this chapter lay the groundwork for the subsequent development and evaluation of the BPredictor4CSL system, paving the way for accurate and efficient sign language segmentation. By addressing the challenges of continuous sign language recognition, this system holds the potential to significantly enhance accessibility and communication for individuals who rely on sign language as their primary mode of expression.

Chapter 4

Results and Discussion

4.1 Introduction

This chapter presents the results and analysis of our experiments on developing a word boundary predictor for continuous sign language using the DGS corpus dataset. We employed a hybrid approach that combined three different machine learning models: Bidirectional Long Short-Term Memory (BiLSTM), Random Forest, and Gradient Boosting. The chapter begins by detailing the optimal hyperparameters obtained through rigorous tuning for each of the three models. Subsequently, we evaluate the performance of the trained models using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, and the Area Under the Receiver Operating Characteristic (AUROC) curve. These metrics provide valuable insights into the models' capabilities, strengths, and limitations in the context of the word boundary prediction task. Additionally, this chapter includes a comparative analysis of the models against baseline algorithms and state-of-the-art techniques. This analysis highlights the improvements achieved by our hybrid approach and discusses the implications of these findings in practical applications. Finally, we explore the potential impact of different preprocessing techniques and the importance of data quality on model performance. This holistic examination provides a thorough understanding of the experimental outcomes and their relevance to advancing continuous sign language recognition systems.

4.2 Development environment

4.2.1 Hardware specifications

In order to implement our solution, we used:

- A laptop computer equipped with an Intel i5-7300HQ processor, operating at 2.50 GHz, 8 GB of RAM, a 256 GB SSD hard drive, and running the Windows 11 Pro operating system was utilized for pre-processing.
- A laptop computer equipped with an Intel i7-6600U, 2.80 GHz and has 8 GB of RAM a 256 GB SSD hard drive, and running the Windows 10 as operating system.
- Another laptop computer equipped with i7-11800H @ 2.30GHz and has 32GB of RAM, a 512GB SSD, and running Linux as main operating system was utilized for training the chosen model on the whole data.
- Google Colaboratory: also known as Colab, is a free and open-source platform introduced by Google. It is specifically designed for developers working on data science applications, particularly those focused on machine learning. Colab provides a user-friendly and collaborative environment for creating, testing, and sharing code, making it an attractive choice for researchers, students, and professionals in the field of data science and artificial intelligence.

4.2.2 Software (Python)

Python is a versatile, high-level programming language known for its simplicity and readability. Developed by Guido van Rossum in the early 1980s[64][65]. Python's multi-paradigm nature and cross-platform compatibility make it a popular choice for a wide range of applications, including operating systems, mobile app development, and video games[66][67]. Python's extensive libraries and active community contribute to its widespread adoption and effectiveness in different domains[64][65][66][67].

4.2.3 Libraries

NumPy: NumPy is a powerful open-source library for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy is widely used in fields such as machine learning, data analysis, and scientific computing [60][62][63].

CV2 (Open-Source Computer Vision): OpenCV (Computer Vision) is an open-source computer vision and machine learning library.
It provides a wide variety of tools for image and video processing, including functions for image manipulation, object detection, and feature extraction. OpenCV is commonly used in applications such as facial recognition, autonomous vehicles, and medical imaging[60][63].

Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting functions, including line plots, scatter plots, bar charts, histograms, and more. Matplotlib is often used in conjunction with other data analysis and machine learning libraries, such as NumPy and Pandas[60][62][63].

Keras: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It is designed to enable fast experimentation with deep neural networks and supports both convolutional networks and recurrent networks, as well as a combination of the two. Keras is widely used in the field of deep learning and is known for its simplicity and ease of use[61].

Tensorflow: TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools, libraries, and community resources that allow developers to build and deploy machine learning-powered applications. TensorFlow is particularly well-suited for large-scale, complex machine learning projects and is used in a wide range of applications, from image recognition to natural language processing[61].

Mediapipe: Mediapipe is an open-source, cross-platform, customizable ML solutions framework for building multimodal applied ML pipelines, including perception and geometry processing solutions. It is widely used in computer vision and robotics applications, such as hand tracking, face detection, and object tracking[60][63].

Flutter Framework: BPredictor4CSL is developed using Flutter, Google's UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase. Flutter's expressive and flexible design language allows for beautiful and highly customizable

user interfaces.

Flask Server: The backend of BPredictor4CSL is powered by a Flask server hosted on Google Colab, a cloud-based development environment. Flask provides a lightweight yet powerful framework for building web applications, handling video uploads, processing tasks, and serving processed results back to the Flutter client.

Ngrok: Ngrok facilitates secure tunneling to expose the locally hosted Flask server to the internet. This enables seamless communication between the Flutter application and the Flask server over HTTPS, ensuring data privacy and security.

4.3 Dataset (DGS corpus)

The DGS (German Sign Language) Corpus is a comprehensive collection of annotated data on German Sign Language, accessible at [1] . The dataset contains almost 356,000 tokens and approximately 50 hours of video footage, updated as of September 2019[48].

The annotations include translations, glosses (similar to Johnston's ID-glosses), and mouthings, with mouth gestures marked by "[MG]" in the Mouthing/Mouth gesture tier[48]. The annotations are provided in both English and German, available through downloadable files with separate tiers for each language[48]. Previous versions of the dataset remain accessible for reference[48]. The DGS Corpus is a significant resource for research on German Sign Language, focusing on the collection and annotation of a large corpus of sign language data[48][49].

Despite the comprehensive nature of the DGS Corpus, we encountered a significant issue with data imbalance, which is common in sign language datasets. Specifically, the boundaries (or transition frames) are significantly fewer compared to the signing frames, leading to an imbalanced dataset. This imbalance posed challenges for developing an effective word boundary predictor, as the minority classes could lead to biased model performance.

To address this, we initially attempted to balance the dataset through data augmentation techniques, specifically by increasing the number of frames at the boundaries. However, this approach proved insufficient to fully mitigate the imbalance. Therefore, we employed advanced resampling techniques within our code to handle the imbalanced dataset. Specifically, we utilized the Adaptive Synthetic Sampling (ADASYN) method to oversample the minority classes. ADASYN dynamically generates synthetic data points for the minority class based on its distribution, which helps in balancing the dataset more effectively and improving model performance. This approach allowed us to enhance the robustness and accuracy of our word boundary prediction models.

4.4 Results

Our models, developed using a hybrid approach were selected based on their superior performance on the validation set, demonstrating strong capabilities in handling the prediction tasks effectively.

[1] http://ling.meine-dgs.de

The BiLSTM model, with its ability to capture long-range dependencies and bidirectional context, demonstrated robust learning and generalization capabilities, making it particularly effective for the sequential data inherent in sign language. The Random Forest model, known for its versatility and robustness in handling diverse datasets, capitalized on its ensemble nature to deliver high accuracy and resilience to overfitting. Meanwhile, the Gradient Boosting classifier leveraged its boosting algorithm to iteratively refine its predictions, improving performance by focusing on the errors of previous iterations.

Each of these models contributed unique strengths to our predictive framework. The BiLSTM excelled in understanding temporal dynamics, the Random Forest provided strong generalization across different feature subsets, and the Gradient Boosting model enhanced precision through its iterative refinement process. By evaluating these models comprehensively, we aimed to harness their individual advantages to achieve optimal performance in predicting word boundaries within continuous sign language data.

This section will delve into the hyperparameters tuned for each model, the detailed performance metrics including accuracy, precision, recall for both classes (0 and 1), and AUPRC scores, and a comprehensive evaluation of their strengths and weaknesses. Through this analysis, we provide a nuanced understanding of the effectiveness of our hybrid model and its suitability for the task at hand.

4.4.1 Hyperparameters

Hyperparameter tuning significantly impacts model performance. We employed an iterative approach,to determine the optimal hyperparameters for each model on our validation set. Tables4.1, 4.2, and 4.3 summarize the final tuned hyperparameters.

For the BiLSTM model, we tuned architectural parameters like layer sizes, dropout rates, and optimizers. The Random Forest and Gradient Boosting models required tuning tree-based parameters like number of estimators, tree depths, learning rates, and minimum sample splits. Proper hyperparameter tuning enhances model performance by balancing complexity and generalization ability to capture data patterns effectively.

- **BiLSTM Model:**

Table 4.1: Hyperparameters of the BiLSTM Model

Parameter Name	Parameter	Value	Description
Number of neurons in the first BiLSTM layer	lstm_units_1	128	Size of the first BiLSTM layer
Number of neurons in the second BiLSTM layer	lstm_units_2	64	Size of the second BiLSTM layer
Dropout rate	dropout_rate	0.5	Dropout rate for regularization
Dense layer neurons	dense_units	64	Size of the dense layer
Kernel regularizer	kernel_regularizer	L2 regularization (0.001)	Regularization technique to prevent overfitting
Batch normalization	batch_norm	Applied after dense layer	Normalization applied to the dense layer
Optimizer	optimizer	Adam (learning rate 0.001)	Optimization algorithm with specified learning rate
Batch size	batch_size	64	Number of samples per gradient update
Number of epochs	epochs	Up to 100 (with early stopping)	Maximum number of training iterations
Loss function	loss_function	Focal Loss (gamma=2.0, alpha=0.25)	Loss function to address class imbalance

- **Random Forest:**

Table 4.2: Hyperparameters of the Random Forest Model

Parameter Name	Parameter	Value	Description
Number of trees	n_estimators	100	Number of trees in the forest
Criterion	criterion	Gini impurity	Function to measure the quality of a split
Max depth	max_depth	None	Maximum depth of the tree
Min samples split	min_samples_split	2	Minimum number of samples required to split an internal node
Min samples leaf	min_samples_leaf	1	Minimum number of samples required to be at a leaf node
Bootstrap samples	bootstrap	True	Whether bootstrap samples are used when building trees
Random state	random_state	42	Seed used by the random number generator

- **Gradient Boosting:**

Table 4.3: Hyperparameters of the Gradient Boosting Model

Parameter Name	Parameter	Value	Description
Number of estimators	n_estimators	100	Number of boosting stages to be run
Learning rate	learning_rate	0.1	Step size shrinkage used to prevent overfitting
Max depth	max_depth	3	Maximum depth of the individual regression estimators
Min samples split	min_samples_split	2	Minimum number of samples required to split an internal node
Min samples leaf	min_samples_leaf	1	Minimum number of samples required to be at a leaf node
Subsample	subsample	1.0	Fraction of samples used for fitting the individual base learners
Random state	random_state	42	Seed used by the random number generator

4.4.2 Performance Metrics of Models

Evaluating the performance of predictive models is essential for understanding their effectiveness in addressing a given task. In this context, we rigorously assess the performance of the best models in our analysis using a diverse set of key metrics such as accuracy, recall, AUPRC, among others. These metrics provide valuable insights into the capabilities of the models, offering a comprehensive understanding of their strengths and limitations.

In this section, we delve into the results of our experiments focused on developing a word boundary predictor within the realm of continuous sign language. Utilizing the DGS corpus dataset and implementing a hybrid approach that combines the predictive abilities of three distinct models, including BiLSTM, Random Forest, and Gradient Boosting, we examine the effectiveness of our methodology in identifying word boundaries within the fluidity of sign language communication.

Classification Report

Table 4.4 presents the classification report, offering detailed insights into the performance of the BiLSTM, Random Forest, and Gradient Boosting models, as well as the combined predictions from these models on the testing set. This report encompasses crucial metrics such as precision, recall, F1-score, and support for each class, enabling a comprehensive evaluation beyond overall accuracy. Additionally, it includes the Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) score and the Area Under the Precision-Recall Curve (AUPRC), indicating the models' ability to distinguish between classes and their precision-recall performance respectively. By examining these metrics, we can assess the strengths and weaknesses of each model, as well as the combined predictions, for this specific dataset, gaining valuable insights into their performance characteristics.

Table 4.4: Classification Report for BiLSTM, Random Forest, and Gradient Boosting Models

Metric	BiLSTM Model	Random Forest Model	Gradient Boosting Model	Combined Prediction
ROC AUC	0.55	0.94	0.97	0.97
AUPRC	0.83	0.98	0.99	0.98
Accuracy	0.57	0.92	0.98	0.98
Precision (class 0)	0.22	0.82	0.99	0.99
Precision (class 1)	0.84	0.95	0.98	0.98
Recall (class 0)	0.50	0.76	0.91	0.91
Recall (class 1)	0.59	0.96	1.00	0.99
F1-Score (class 0)	0.30	0.79	0.95	0.95
F1-Score (class 1)	0.69	0.95	0.99	0.99
Macro Avg Precision	0.53	0.88	0.98	0.98
Macro Avg Recall	0.54	0.86	0.95	0.96
Macro Avg F1-Score	0.50	0.87	0.97	0.97
Weighted Avg Precision	0.72	0.92	0.98	0.98
Weighted Avg Recall	0.57	0.92	0.98	0.98
Weighted Avg F1-Score	0.62	0.92	0.98	0.98
Support	2422	2422	2422	2422

The BiLSTM model exhibited relatively poor performance, achieving an accuracy of 0.57, suggesting its limited capability to learn and generalize from the training data effectively. The Random Forest and Gradient Boosting models outperformed the BiLSTM model, attaining superior accuracies of 0.92 and 0.98, respectively, demonstrating their suitability for the given dataset.

While the individual models displayed varying levels of performance, the combined predictions from all three models emerged as the best results. The combined predictions achieved an accuracy of 0.98, outperforming the individual models. Additionally, the combined predictions yielded impressive metric scores, including a precision of 0.99 for class 0 and 0.98 for class 1, a recall of 0.91 for class 0 and 0.99 for class 1, and F1-scores of 0.95 and 0.99 for classes 0 and 1, respectively.

The combined predictions also demonstrated strong discrimination ability, with an ROC AUC of

0.97 and an AUPRC of 0.98, comparable to the best individual model (Gradient Boosting). The macro averages for precision, recall, and F1-score were 0.98, 0.96, and 0.97, respectively, while the weighted averages for precision, recall, and F1-score were 0.98, 0.98, and 0.98, respectively. By leveraging the strengths of multiple models and mitigating their individual weaknesses, the combined predictions outperformed the individual models, achieving the best overall performance in terms of accuracy, precision, recall, F1-score, and discrimination ability (ROC AUC and AUPRC). This highlights the effectiveness of ensemble techniques in improving classification performance and underscores the superiority of the combined predictions in this evaluation.

Confusion Matrix

The confusion matrix (Table 4.5) presents the prediction results of three distinct models: Random Forest (RF), Gradient Boosting (GB), and BiLSTM Model, as well as the combined predictions from these models. For the Random Forest model, the confusion matrix shows better performance, correctly predicting 1899 instances of class 1, but misclassifying 109 instances of class 0 as class 1 (false positives) and 74 instances of class 1 as class 0 (false negatives). The Gradient Boosting model exhibited the best individual performance, correctly predicting 1968 instances of class 1 and 408 instances of class 0, with 41 false positives for class 0 and 5 false negatives for class 1.

The BiLSTM Model, on the other hand, demonstrated relatively poor performance, correctly predicting only 1162 instances of class 1 and 224 instances of class 0, while misclassifying 225 instances of class 0 as class 1 and 811 instances of class 1 as class 0.

However, when the predictions from all three models were combined, the resulting confusion matrix showed the best overall performance. The combined models accurately classified 1968 instances of class 1 and 410 instances of class 0, with 39 false positives for class 0 and only 5 false negatives for class 1. This improvement in the combined predictions can be attributed to the ensemble effect, where the strengths of individual models are leveraged, and the weaknesses are mitigated. By combining the predictions, the ensemble model benefited from the collective knowledge and diverse perspectives of the individual models, leading to superior overall performance compared to any single model.

Therefore, the confusion matrix analysis suggests that the combined predictions from the Random Forest, Gradient Boosting, and BiLSTM models provide the best classification results, effectively minimizing misclassifications and achieving the highest accuracy among the evaluated scenarios.

Table 4.5: Confusion Matrix for BiLSTM, Random Forest, Gradient Boosting, and Combined Models

Model	Actual	Predicted: 0	Predicted: 1
Random Forest	0	340	109
	1	74	1899
Gradient Boosting	0	408	41
	1	5	1968
BiLSTM	0	224	225
	1	811	1162
Combined	0	410	39
	1	5	1968

4.4.3 Comparison Results(Test Models on DGS corpus Dataset

In this section, we present the testing results of various models evaluated on the DGS dataset. The metrics highlighted focus on the models' performance when applied to the testing data, which consists of unseen samples not used during training. Refer to Table 4.6 for a comprehensive overview and comparison of the testing results of the different models across multiple evaluation metrics.

Table 4.6: Performance Evaluation of Various Models on DGS Dataset

Model	Average Time prediction per frame (ms)	Accuracy	Recall (Class 0)	Recall (Class 1)	Precision (Class 0)	AUPRC	ROC AUC
1D CNN 1conv1D with attention	0.32	0.73	0.0	1.0	0.0	0.71	0.47
1D CNN 3conv1D + attention	0.37	0.73	0.0	1.0	0.0	0.72	0.50
XGBoost + RandomForest Stacked with 1D CNN	0.77	0.71	0.03	0.97	0.23	0.73	0.49
XGBoost, RandomForest with Weighted 1D CNN	0.36	0.71	0.02	0.97	0.20	0.72	0.49
Stacked Ensemble with Focal Loss 1D CNN	0.82	0.31	0.92	0.09	0.27	0.73	0.50
Anomaly Detection CNN Model	0.45	0.66	0.14	0.85	0.26	0.72	0.49
DNN with ADASYN Resampling	0.76	0.38	0.79	0.22	0.27	0.73	0.51
DNN with ADASYN Resampling and Cross Validation	0.58	0.37	0.82	0.20	0.27	0.74	0.51
Sequential with ADASYN	0.83	0.40	0.57	0.33	0.24	0.71	0.45
BiLSTM with Focal Loss and ADASYN	0.57	0.57	0.50	0.59	0.22	0.84	0.56
Bidirectional LSTM with RF	0.63	0.41	0.98	0.28	0.24	0.98	0.94
Bidirectional LSTM with GB	0.58	0.72	0.95	0.67	0.40	0.99	0.97
Bidirectional LSTM with GB and RF	**0.64**	**0.98**	**0.91**	**0.99**	**0.99**	**0.98**	**0.97**

After evaluating these models, we found that none provided satisfactory results for both classes. As illustrated in Table 4.6, many models only predicted one class correctly or showed poor performance for both classes.
However, the BiLSTM model, despite its overall poor results, achieved a balanced performance with an average recall of about 50% for both classes. This indicates that the BiLSTM model can predict the two classes relatively equally, even though its overall accuracy is not high.
When we observed that the BiLSTM model, despite its poor overall performance, achieved a more balanced prediction for both classes, we attempted to leverage this by integrating it with other models. It was found that combining the BiLSTM with machine learning models, specifically Random Forest (RF) and Gradient Boosting (GB), resulted in the best overall performance. The integration of BiLSTM with RF and GB enhanced the model's ability to handle the imbalanced dataset, providing more accurate and reliable predictions.

4.5 Comparison Results with Model Proposed in Literature

The following section presents a detailed comparison of various models evaluated on the DGS dataset, highlighting their performance using different feature sets and classification techniques, as shown in Table 4.7.

Table 4.7: Performance Comparison of DGS dataset

Classifier	Feature	Accuracy	Year	Reference
BiLSTM + GB + RF	**64 landmarks (Face and Hands)**	**98**	**2024**	**our work**
LSTM (Pose-All)	137 landmarks (face, body, hands)	91.53	2020	[68]
LSTM (Pose-Body)	25 body landmarks	90.35	2020	[68]
LSTM (Pose-Hands)	42 hand landmarks (21 each hand)	88.05	2020	[68]
LSTM (BBOX)	8 bounding box features (face, body, hands)	87.78	2020	[68]
Linear-25	25 optical flow features with context window of 25 frames	83.79	2020	[68]
Linear-50	25 optical flow features with context window of 50 frames	83.39	2020	[68]
Linear-1	25 optical flow features	79.93	2020	[68]

The Linear-1 model uses 25 optical flow features, achieving an accuracy of 79.93%. Expanding this to include 25 frames, the Linear-25 model improves accuracy to 83.79%. However, further increasing to 50 frames, the Linear-50 model experiences a slight decrease in accuracy to 83.39%. These results suggest that while increasing the number of frames generally enhances performance, linear models have limitations in capturing the complex temporal dynamics present in sign language data.

The LSTM models, on the other hand, leverage various landmark features to improve performance. The LSTM (BBOX) model uses 8 bounding box features (face, body, hands) and achieves an accuracy of 87.78%. Focusing on hand landmarks, the LSTM (Pose-Hands) model, which uses 42 hand landmarks, slightly improves accuracy to 88.05%. When using 25 body landmarks,

the LSTM (Pose-Body) model significantly boosts accuracy to 90.35%. Combining all landmarks (137 in total, covering face, body, and hands), the LSTM (Pose-All) model reaches the highest accuracy of 91.53% among the LSTM models. This progression in accuracy with the inclusion of more comprehensive landmark features underscores the importance of detailed spatial information in enhancing model performance.

Our proposed hybrid model (BiLSTM + RF + GB) significantly outperforms all previously mentioned models, achieving an impressive accuracy of 98%. The key factors contributing to this superior performance are the unique strengths of each component model.

- **Bidirectional Long Short-Term Memory (BiLSTM)** networks are highly effective for sequence prediction tasks, processing data in both forward and backward directions to capture dependencies across the entire sequence. This bidirectional approach is particularly beneficial for understanding complex temporal patterns within the sign language data.
- **Random Forest (RF)** adds robustness by aggregating the predictions of multiple decision trees, reducing the risk of overfitting. This model is highly effective in handling a variety of data features and improving the generalization capability of the model.
- **Gradient Boosting (GB)** enhances the model's accuracy by sequentially building trees that correct the errors of previous trees. This iterative improvement process results in a strong predictive model capable of fine-tuning its performance based on the intricacies of the dataset.

The combination of these models in a hybrid system allows us to capitalize on the strengths of each while mitigating their individual weaknesses, resulting in a significant performance boost compared to standalone models.

4.6 Developed Tools: BPredictor4CSL

BPredictor4CSL is a sophisticated Flutter-based mobile application designed for efficient video uploading, processing, and playback. Developed to streamline the interaction between users and server-side processing capabilities, BPredictor4CSL offers a seamless interface combined with robust functionality.

4.6.1 Application Interface

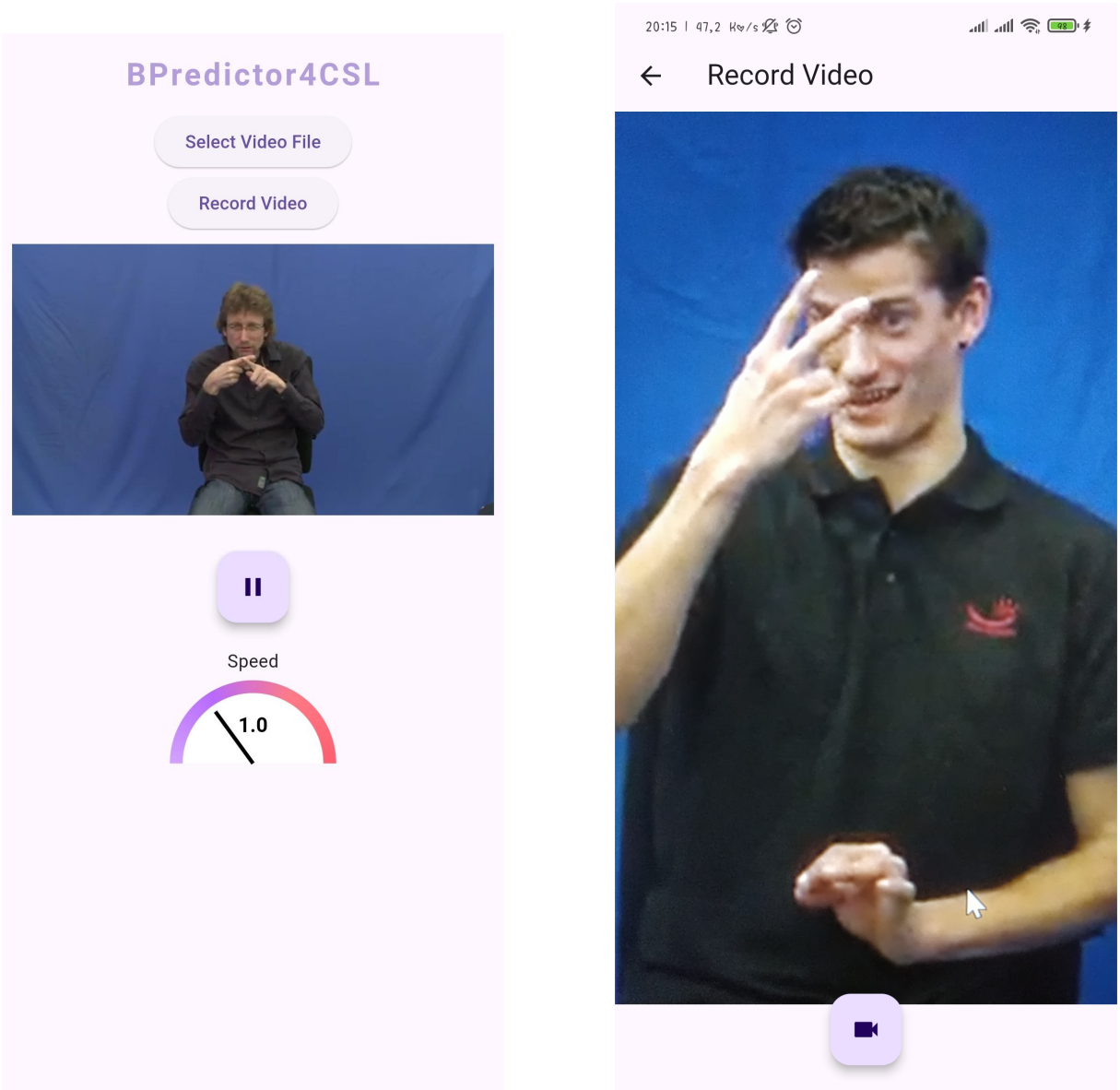
The interface of BPredictor4CSL is intuitive and user-friendly, featuring essential controls for video management and playback. Upon launching the application, users are greeted with a straightforward layout that emphasizes usability, as shown in Figure4.1:

- **Video Upload**

  Users can initiate the video upload process in two ways:

1. **Select Video File:** Clicking this button (replace with image of "Select Video File" button), as illustrated in Figure 4.1(a) prompts the selection of a pre-recorded video file from the device's storage.
2. **Record Video:** Alternatively, users can click the "Record Video" button (replace with image of "Record Video" button) to directly capture a new video using the device's camera, as shown in Figure 4.1(b). This provides the flexibility to upload fresh footage without needing to pre-record it.

- **Video Display**: Once a video is selected, BPredictor4CSL immediately displays the chosen video using a built-in VideoPlayer component. This allows users to preview the video content directly within the application.



((a)) Upload video ((b)) Record video

Figure 4.1: Graphical interfaces of the input

- **Processing and Playback**: Behind the scenes, BPredictor4CSL leverages a Flask server hosted on Google Colab, accessible via Ngrok for secure communication. Upon uploading a video, the application sends it to the Flask server for sophisticated processing. This includes tasks such as video analysis, transformations, and filtering based on predefined algorithms.
- **Processed Video Playback**: After processing is completed on the server, BPredictor4CSL retrieves the processed video and seamlessly transitions to displaying the enhanced or modified video content. Users can toggle playback, replay processed videos, and adjust playback speed for a customized viewing experience, as shown in Figure 4.2.

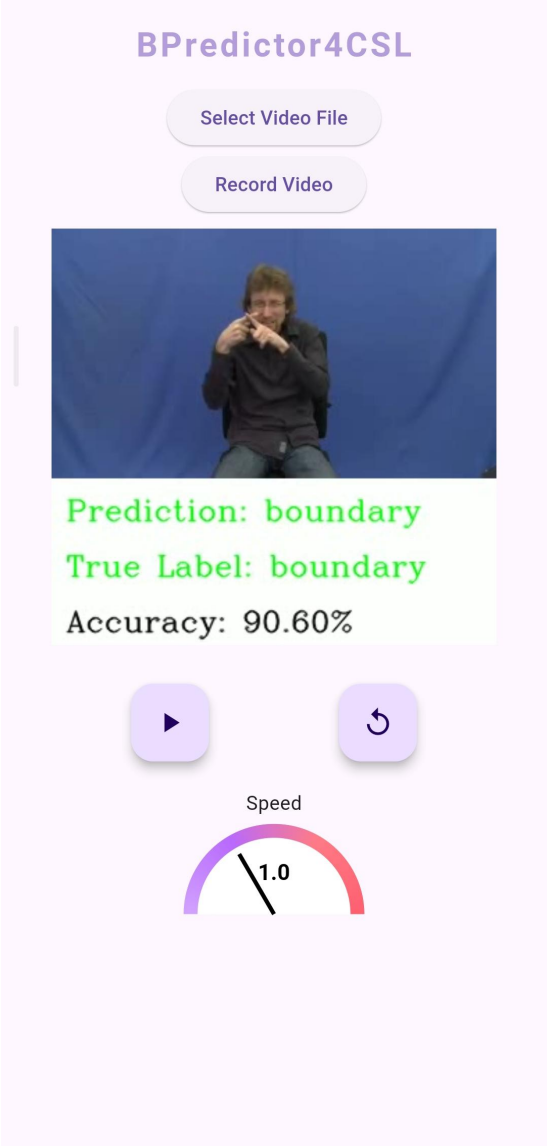


Figure 4.2: Graphical interface of the output

4.6.2 Key Functionalities

BPredictor4CSL incorporates several key functionalities that enhance the user experience and optimize video processing tasks:

1. **Video Selection and Upload**:
    - Users can select any video file from their device for processing or they can record a new video.
    - BPredictor4CSL handles the upload process seamlessly, ensuring efficient data transfer to the Flask server.
2. **Server-side Processing**:
    - Utilizing Flask and Ngrok technologies, the application communicates securely with the server for video processing tasks.
    - Advanced algorithms execute operations such as video enhancement and data extraction, enhancing the video content based on predefined parameters.
3. **Interactive Playback Controls**:
    - BPredictor4CSL provides users with interactive playback controls, including play/pause functionality for both original and processed videos.
    - A speed control gauge allows users to adjust playback speed, providing flexibility in viewing the content at their preferred pace.
4. **Error Handling and User Feedback**:
    - The application incorporates robust error handling mechanisms to notify users of any upload or processing failures.
    - Informative dialog boxes guide users through potential issues, ensuring a smooth and frustration-free user experience.

BPredictor4CSL exemplifies a fusion of innovative mobile application design and backend server capabilities. By integrating advanced video processing functionalities with an intuitive user interface, the application enhances user engagement and productivity. Whether for personal use or professional applications, BPredictor4CSL sets a new standard in mobile video management and enhancement.

4.7 Conclusion

In this chapter, we presented the results and analysis of our experiments focused on developing an effective word boundary predictor for continuous sign language using the DGS corpus dataset.

Our approach employed a hybrid model that combined the strengths of three distinct machine learning techniques: Bidirectional LSTM, Random Forest, and Gradient Boosting. We discussed the development environment, hardware specifications, and software requirements utilized in our implementation. Through rigorous hyperparameter tuning and evaluation, we demonstrated that the combined predictions from all three models outperformed any single model, attaining an accuracy of 98%, along with impressive scores across other critical metrics such as precision, recall, F1-score, ROC AUC, and AUPRC. These results underscore the effectiveness of ensemble learning techniques in improving classification performance, particularly in the context of challenging tasks like continuous sign language recognition. Additionally, we compared our model's performance with other state-of-the-art approaches on the same DGS dataset, highlighting the advantages of our methodology.

General Conclusion

This study embarked on the development and evaluation of the BPredictor4CSL system, a cutting-edge mobile application designed to predict segment boundaries in continuous sign language videos. By utilizing advanced deep learning techniques, particularly Bidirectional Long Short-Term Memory (BiLSTM) networks, alongside machine learning models like Random Forest (RF) and Gradient Boosting (GB), we achieved substantial improvements in the accuracy and efficiency of sign language recognition. Our research aimed to bridge the communication gap between the deaf community and non-deaf individuals, thereby promoting inclusivity.

The research methodology was comprehensive, encompassing multiple phases including data preparation, preprocessing, model training, and evaluation. Key steps involved video frame extraction, facial and hand landmark detection, and feature extraction, which collectively contributed to building a robust dataset for training our predictive models. The BPredictor4CSL mobile application was rigorously evaluated using standard metrics such as confusion matrices, classification reports, AUPRC, and ROC AUC scores. The results indicated that the system performed well across these metrics, highlighting its potential to significantly improve communication for individuals who rely on sign language.

The findings of this study underscore the transformative potential of integrating deep learning technologies into sign language recognition systems. By enhancing the accuracy and efficiency of sign language segmentation, the BPredictor4CSL mobile application can facilitate more effective communication and interaction between the deaf community and the broader population. This advancement not only supports social inclusion but also sets a foundation for further innovations in this field.

Future Works

To build on the promising results of this study, future research could explore the following areas:

- **Model Optimization**: Further optimization of the BiLSTM, Random Forest, and Gradient Boosting models to enhance their performance and reduce computational complexity.
- **Dataset Expansion**: Incorporating a larger and more diverse dataset to improve the generalizability of the BPredictor4CSL system across different sign languages and dialects.

- **Real-time Processing**: Developing and testing real-time processing capabilities to enable instant translation and interpretation of sign language in live interactions.
- **User Interface Enhancement**: Improving the user interface and experience of the BPredictor4CSL mobile application to make it more accessible and user-friendly for a wider audience.
- **Cross-disciplinary Applications**: Investigating the application of the system in other areas such as sign language education, virtual reality, and human-computer interaction.
- **Ethical and Societal Impact**: Conducting studies to understand the ethical implications and societal impact of deploying sign language recognition systems, ensuring that they are used responsibly and equitably.

These future directions will help to further refine and expand the capabilities of the BPredictor4CSL mobile application, contributing to the ongoing effort to bridge communication gaps and foster inclusivity for the deaf community.

Bibliography

[1] Pramila P. Shinde and Seema Shah. (August 2018). *A Review of Machine Learning and Deep Learning Applications*, 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

[2] Lei Zhang, Shuai Wang, and Bing Liu. (March 30, 2018). *Deep Learning for Sentiment Analysis*: *A Survey*, WIREs Data Mining and Knowledge Discovery, Vol. 8, No. 4.

[3] Mayank Banoula. (April 27, 2023). *Introduction to Long Short-Term Memory* (*LSTM*): *Simplilearn*, Simplilearn.com.

[4] Simplilearn, (August 10, 2023). *What Are Autoencoders*? *Introduction to Autoencoders in Deep Learning*, Simplilearn.com.

[5] Avijeet Biswal. (February 15, 2024). *Top* 10 *Deep Learning Algorithms You Should Know in* 2024, Simplilearn.com.

[6] Arden Dertat. (October 8, 2017). *Applied Deep Learning - Part* 3: *Autoencoders*, Medium.

[7] Matteo Lionello and Hendrik Purwins. (November 19, 2020). *End-to-End Dilated Variational Autoencoder with Bottleneck Discriminative Loss for Sound Morphing – a Preliminary Study*, arXiv.org.

[8] Aqeel Anwar. (November 4, 2021). *Difference between AutoEncoder* (*AE*) *and Variational AutoEncoder* (*VAE*), Medium.

[9] Jérémy Robert. (March 14, 2024). *Intelligence Artificielle*: *Définition*, *Histoire*, *Enjeux*, Formation Data Science | DataScientest.com.

[10] Nicole Laskowski and Linda Tucci. (November 13, 2023). *What Is Artificial Intelligence and How Does AI Work*?: *Definition from TechTarget*, Enterprise AI.

[11] Oracle France, *Qu'est-Ce Que Le Machine Learning*?, Qu'est-ce que le machine learning? | Oracle France.

[12] Julianna Delua. (March 12, 2021). *Supervised vs. Unsupervised Learning*: *What's the Difference*?, IBM Blog.

[13] Dr. Sunil Kumar Chinnamgari. (2019). *R Machine Learning Projects*: *Implement Supervised, Unsupervised, and Reinforcement Learning Techniques Using R* 3.5, Packt Publishing Ltd. ebin.pub.

[14] IBM, *What Are Recurrent Neural Networks?*, IBM.

[15] Mishra Mayank. (September 2, 2020). *Convolutional Neural Networks, Explained*, Medium, Medium.

[16] Baheti Pragati. (June 27, 2021). *A Comprehensive Guide to Convolutional Neural Networks*, V7 Labs. https://www.v7labs.com.

[17] IBM, (2024). *What Is a Neural Network?*, IBM.

[18] *Weights and Biases*, n.d. Last modified 2 years ago. Weights and Biases - AI Wiki.

[19] Ognjanovski, Gavril. *Everything You Need to Know about Neural Networks and Backpropagation-Machine Learning Easy and Fun*, Medium, June 7, 2020. Medium.

[20] Yathish, Vishal. (August 4, 2022). *Loss Functions and Their Use in Neural Networks*. Medium. Medium.

[21] Oppermann Artem. *How Loss Functions Work in Neural Networks and Deep Learning*. Built In, n.d. Built In.

[22] *Deep Learning* : *Les Réseaux de Neurones Récurrents*. DataValue Consulting part of Mazars, August 25, 2021. DataValue Consulting part of Mazars.

[23] Machine Learning in Plain English. (June 21, 2023). *Convolutional Neural Network - Lesson* 8: *Fully Connected Layers and Flattening*. Medium. Medium.

[24] *Fully Connected Layers in Convolutional Neural Networks*. IndianTechWarrior, January 24, 2024. IndianTechWarrior.

[25] KOUAR Imene, KOUAR El Bachir, KHELDOUN Ahmed. (2023). *Deep-RSL*: *Deep Learning for Sign Language Recognition from a Video Sequence*. Catalogue en ligne. Catalogue en ligne.

[26] *Neural Networks*. StudySmarter UK, n.d. StudySmarter UK.

[27] Raj Ravish. *Components of an Artificial Neural Network*. enjoyalgorithms, n.d. enjoyalgorithms.

[28] Britta Bauer and Hermann Hienz. "Relevant Features for Video-Based Continuous Sign Language Recognition." *Proceedings of the Aachen University of Technology* (*RWTH*), Department of Technical Computer Science, Ahornstr. 55, 52074 Aachen, Germany, email: {fbauer, hienz}@techinfo.rwth-aachen.de.

[29] Cooper, R., Dobrushina, E., Watt, S., Alexander, G., Bowden, R., Crossley, S., Cuddihy, E., Dufour, L., Garrette, D., Greenham, D., et al. (2013). New Features in Vireo 2.0. arXiv preprint arXiv:1306.2519.

[30] Cormier, K., & Schembri, A. (2021). *Sign Languages*: *A Comparative Perspective*. Brill.

[31] Emmorey, K. (2003). *The Oxford Handbook of Deaf Studies in Language and Education*. Oxford University Press.

[32] Liddell, S. K. (2003). *Grammar and the Lexicon of New Zealand Sign Language*. Walter de Gruyter.

[33] Word Boundary Detection Using a Conditional Random Field Model. In *Proceedings of the Association for Computational Linguistics*, 2006.

[34] Neural Network-Based Word Boundary Detection for Continuous Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 2018.

[35] Word Boundary Prediction for Continuous Speech Recognition. In *Interspeech*, 2006.

[36] Dang Khanh and Igor A. Bessmertny, "Segmentation of word gestures in sign language video," *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:265238283`.

[37] Zhenxing Zhou, Vincent W. L. Tam, and Edmund Y. Lam, "A Cross-Attention BERT-Based Framework for Continuous Sign Language Recognition," *IEEE Signal Processing Letters*, vol. 29, pp. 1818-1822, 2022. [Online]. Available: `https://api.semanticscholar.org/CorpusID:251653465`.

[38] Chengcheng Wei, Jian Zhao, Wen-gang Zhou, and Houqiang Li, "Semantic Boundary Detection With Reinforcement Learning for Continuous Sign Language Recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, pp. 1138-1149, 2021. [Online]. Available: `https://api.semanticscholar.org/CorpusID:226553562`.

[39] Necati Cihan Camgöz, Simon Hadfield, Oscar Koller, and R. Bowden, "SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition," 2017 *IEEE International Conference on Computer Vision* (*ICCV*), pp. 3075-3084, 2017. [Online]. Available: `https://api.semanticscholar.org/CorpusID:38408000`.

[40] Jens Forster, Christian Oberdörfer, Oscar Koller, and Hermann Ney, "Modality Combination Techniques for Continuous Sign Language Recognition," *Iberian Conference on Pattern Recognition and Image Analysis*, 2013. [Online]. Available: `https://api.semanticscholar.org/CorpusID:11640638`.

[41] Gangqing Yu and Tiantian Yuan, "SkeFormer: Sign Language Segmentation Based on Skeleton Input Transformer," 2023 *IEEE 3rd International Conference on Electronic Technology, Communication and Information* (*ICETCI*), pp. 298-301, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:259970474`.

[42] ALGSL89: An Algerian Sign Language Dataset, KHELDOUN, Ahmed and KOUAR, Imene and KOUAR, El Bachir, 2023 , Uluslararası Yönetim Bilişim Sistemleri ve Bilgisayar Bilimleri Dergisi.

[43] Eesham Fatima, Waqasia Naeem, and Irfan Abbas. THE INFLUENCE OF GENDER ON THE DISCOURSE MARKERS IN PAKISTANI SIGN LANGUAGE. *Pakistan Journal of Social Research*, 2022. URL: https://api.semanticscholar.org/CorpusID:253471669.

[44] Sarah Ebling, Reiner Konrad, Penny Boyes Braem, and Gabriele Langer. Factors to Consider When Making Lexical Comparisons of Sign Languages: Notes from an Ongoing Comparison of German Sign Language and Swiss German Sign Language. *Sign Language Studies*, 16:30–56, 2015. URL: https://api.semanticscholar.org/CorpusID:143044344.

[45] J.M. Power, G.W. Grimm, and J.M. List. Evolutionary dynamics in the dispersal of sign languages. *R Soc Open Sci*, 7(1):191100, 2020. DOI: 10.1098/rsos.191100.

[46] Diane Lillo-Martin and Jon Henner. Acquisition of Sign Languages. *Annual Review of Linguistics*, 7:395–419, 2021. DOI: 10.1146/annurev-linguistics-043020-092357.

[47] Martha Kaperotxipi Urain. Sign language phonology and the system. In *Proceedings of the ...*, 2006. URL: https://api.semanticscholar.org/CorpusID:59737897.

[48] *Willkommen beim öffentlichen DGS-korpus release* 3! *Hintergrundinformationen*, (n.d.), `https://www.sign-lang.uni-hamburg.de/meinedgs/ling/start_de.html`.

[49] Thomas Hanke. (2008), *DGS Corpus Project – Development of a Corpus Based Electronic Dictionary German Sign Language / German*, `https://www.semanticscholar.org/paper/182beebc6f630b0021a0eab639bde31ab12ec48c`.

[50] Kundu, R. (2022, September 13), *Confusion matrix: How to use it & interpret results* [*examples*]. `https://www.v7labs.com/blog/confusion-matrix-guide`

[51] AlmaBetter.(n.d.), *Metrics for classification model*. `https://www.almabetter.com/bytes/tutorials/data-science/classification-metrics`

[52] Kumar S,(2023, January 9), *8important evaluation metrics for classification models*. AITUDE. `https://www.aitude.com/8-important-evaluation-metrics-for-classification-models/`

[53] Tripathi,A.S.(2022, October 13), *Classification metrics and their use cases*. CloudxLab Blog. `https://cloudxlab.com/blog/classification-metrics/`

[54] Joseph Robinson, Ph. D. (2023, December 1), *Unveiling the secrets of log-loss*. Medium. `https://towardsdatascience.com/secrets-of-log-loss-84c668f4024a`

[55] Dyakonov, A. (2021, February 15), *Log loss function* . https://dasha.ai/en-us/blog/log-loss-function

[56] Dembla, G. (2020, November 17), *Intuition behind log-loss score*. Medium. https://towardsdatascience.com/intuition-behind-log-loss-score-4e0c9979680a

[57] Odmark, J. (2022, November 14), *Defining log loss in machine learning applications*. Pandio. https://pandio.com/what-is-log-loss-in-machine-learning/

[58] Brownlee, J. (2020, September 15). *ROC curves and precision-recall curves for imbalanced classification*. MachineLearningMastery.com

[59] Draelos, R. (2024, January 19). *Measuring performance*: *AUPRC and average precision*. Glass Box

[60] Faes, D. M. (2018, July 12). *Use of python programming language in astronomy and science*. arXiv.org.

[61] Ott, J., Pritchard, M., Best, N., Linstead, E., Curcic, M., & Baldi, P. (2020, August 4). *A Fortran-Keras Deep Learning Bridge for scientific computing*. arXiv.org.

[62] Kumar, R. (2015,September 31). *Future for scientific computing using Python*. International Journal of Engineering Technologies and Management Research. https://www.granthaalayahpublication.org/.

[63] Zhang, H., Yang, P., & Niu, Y. (2023). *Application of Python Scientific computing library and Simulation in Circuit Analysis*. 2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT), 892-898. https://ieeexplore.ieee.org/document/10134600.

[64] Guido van Rossum . 2000. *Python Reference Manual*, https://www.semanticscholar.org,

[65] Faes, D. M. (2018). *Use of Python programming language in astronomy and science*. ArXiv,

[66] Kumar, R. (2015, July 31). *Future for scientific computing using Python*. International Journal of Engineering Technologies and Management Research. https://www.granthaalayahpublication.org/ijetmr-ojms/index.php/ijetmr/article/view/04_IJETMR15_A07_22

[67] Pollard, T. J., Johnson, A. E. W., Raffa, J. D., & Mark, R. G. (2018, May 23). *tableone*: *An open source python package for producing summary statistics for research papers*. JAMIA open. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6951995/

[68] Moryossef, A., Tsochantaridis, I., Aharoni, R., Ebling, S., & Narayanan, S. (2020). *Real-time sign language detection using human pose estimation*. Computer Vision – ECCV 2020 Workshops, 237–248. https://doi.org/10.1007/978-3-030-66096-3_17