Chapter 2

Related Work and Preliminaries

The literature review was performed with a focus on EMG-based HGR models. We grouped the literature review based on static and dynamic gestures for better understanding. The initial focus was on EMG applications using static gestures, and then systematically, it moved towards applications based on dynamic hand gesture recognition models. For static hand gestures, we primarily reviewed research papers on sign language gesture recognition, particularly American Sign Language (ASL). ASL, one of the most widely used sign languages globally, includes 26 letters and ten digits (fingerspelling), which are predominantly conveyed through static gestures [38]. Fingerspelling is a crucial aspect of sign language communication, allowing users to spell out words that cannot be easily represented by standard signs.

For dynamic gestures, we consider hand movements involved in writing English alphabets. Research articles on EMG-based recognition of handwritten English alphabets were analyzed and comprehensively reviewed.

The review primarily focuses on machine learning-based systems for gesture recognition using EMG. During which we identified the common structure of these

HGR systems, analyzed the existing models in the literature, and cover the standardized concepts across various aspects, including model types, data acquisition, segmentation, preprocessing, feature extraction, classification, postprocessing, real-time processing, gesture types, and evaluation metrics. Additionally, we identify trends and gaps that could suggest new directions for future research in this area. This analysis provides a comprehensive overview of the current state-of-the-art of these fields and highlights potential areas for further investigation.

Our review identifies the limitations of existing methods and highlights potential areas for future research. Additionally, Section 2.3.3 provides an overview of the datasets employed for developing both static and dynamic hand gesture models. This chapter also includes the preliminary concepts that are frequently employed in our experiments.

2.1 EMG-based Sign Language Gesture Recognition

Sign languages are considered visual and non-verbal forms of communication used by differently-abled people to express themselves or interact with their surroundings. These languages are expressed using manual and non-manual features. These features mainly include different hand movements, use of different numbers of fingers, palm orientation, facial expression, head orientation, hand shape, eye gaze, etc. [39] [40]. American Sign Language (ASL) is one of the most widely used sign languages comprising various static gestures as a means of human expression [41].

The majority of work on American Sign Language (ASL) recognition work can be classified into two broad categories based on data acquisition: visual and sensor-based approaches. Visual approaches use camera input data and apply image processing algorithms to build a sign recognition system. Meanwhile, sensor-based

approaches gather data from various devices, such as gloves, wearable sensors, accelerometers, gyroscopes, and surface electromyography sensors [5, 42].

The development of visual sign language recognition began in the 1980s, with a focus on features derived from hand, face, and body poses [42]. With the advent of machine learning, researchers began using neural networks and image processing techniques to recognize American Sign Language (ASL) gestures [43] [44]. Building on the success of deep learning models, Convolutional Neural Networks (CNNs) were later adopted to develop more accurate Sign Language Recognition Systems (SLRS) [45–47]. SLRS are technological solutions designed to interpret and translate sign language gestures into spoken or written language [5]. These systems can assist, automate, act as translators, and enhance interaction between sign language users and those who do not understand sign language, thereby promoting inclusivity and accessibility. While CNNs were effective for static image frames, they struggled with sequence information in continuous frames [48]. To address this issue, hybrid models combining CNNs with Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN) were proposed [49–53].

The efficiency of visual-based ASL SLRS was further enhanced by depth cameras and accurate motion trackers like Kinetic sensors and Leap Motion sensors [4]. Kinetic sensors track whole-body movements, while Leap Motion sensors are specialized in precise hand gestures [54]. ASL recognition systems using these sensors achieved high classification accuracy, such as 79.83% with Support Vector Machine [55] and 99.44% with Recurrent Neural Networks [51].

Kinetic sensors contribute to about 20% of SLRS research, capturing motion and depth information to build robust, multi-modal systems [56]. However, visual and depth-based systems face challenges due to variables like camera viewpoint, resolution, lighting conditions, and the need for a clear line of sight [57, 58].

Sensor-based methods, particularly sensory gloves, have also proven effective. These gloves, equipped with various sensors, monitor the physical features of ASL

gestures [59]. Notable studies have achieved up to 96% accuracy using sensory gloves combined with neural networks [60–65]. However, these gloves can be expensive and cumbersome for real-time use, as they cover most of the user's palm and fingers, restricting other tasks.

An emerging and cost-effective alternative is wearable sEMG (surface electromyography) sensors. sEMG-based methods provide a viable alternative for developing accurate and efficient ASL recognition systems. These methods are less affected by lighting conditions and other visual constraints, making them suitable for real-time applications [66]. Research on sEMG-based methods for ASL recognition has shown promising results. For instance, Savur et al. [67] applied a Myo armband with eight sEMG channels and extracted various time and frequency domain features from raw sEMG signals. Using baseline machine learning classifiers, achieved a classification accuracy of 61.04%. In the thesis work of Jackson Taylor [68], three different sensors, viz. accelerometer, gyroscope, and sEMG sensors (total 15 different channels), were applied for classifying different ASL gestures. Using k-NN and dynamic time windowing, the author achieved an accuracy in the range of 94%-98%. Similarly, Paudyal et al. [69] applied two wrist-worn devices consisting of sEMG and inertial sensors to recognize the ASL signs using Dynamic Time Wrapping, an energy-based approach. By combining all the sensors (34 different channels), Paudyal et al. achieved an accuracy of 97.72%. Wu et al. [70] proposed an ASL recognition system using the sEMG and wrist-worn inertial sensors. They obtained the recognition accuracy of 95.94%, Using three different sensors with a total of ten channels (6 inertial sensor channels with four sEMG channels). An information gain-based filter method was applied to select the 30 best features set from 268 features. With the extension of the previous work, the authors applied an auto-segmentation technique to identify the occurrence of ASL gesture [71]. Using additional gyroscope channels to the feature vector, the proposed method achieved a classification accuracy of 96.16%. In [72], while performing sEMG-based ASL gesture recognition, Kosmidou et al. applied Discriminant analysis to find the best features subset from the total

number of features extracted. The authors claim to achieve a classification accuracy of 97.7%. In [73], several Time domain features were extracted for the sEMG channels corresponding to different ASL gestures. Using a Support vector machine, their model achieved an offline accuracy of 91.1%. No feature selection method was applied to reduce the feature vector. Their proposed model uses eight sEMG channels, and its performance was evaluated on 2080 samples. Fatmi et al. [66] used two Myo Armbands with sEMG and inertial sensors, totaling 26 channels, to develop an ASL recognition model. They compared the performance of Artificial Neural Networks (ANN) and Support Vector Machines (SVM) with the Hidden Markov Model (HMM), achieving classification accuracies of 93.79% and 89.05%, respectively. Although many sEMG-based ASL recognition models employ multiple sensors to boost accuracy, this increases the system's cost. Furthermore, there has been minimal effort to identify optimal methods for achieving high classification accuracy with fewer sensors, highlighting a significant limitation in current research. Table 2.1 summarizes the related work in this area.

TABLE 2.1: Summary of the related work for sEMG sensor based ASL recognition

References	Various sensor used	No of samples	No of gestures	No of subjects	No of channels	Classification Accuracy
[67]	1	1040	26	10	8	61.04%
[66]	4	26000	13	3	26	93.79%
[69]	3	-	20	10	34	97.72%
[70]	3	3000	40	4	10	95.94%
[71]	3	24000	80	4	10	85.24%-96.16%
[72]	1	180	9	-	2	97.7%
[73]	1	2080	26	-	8	91.1%
[68]	3	-	20	-	15	94%-98%

2.2 EMG-based handwritten character gesture recognition

Despite rapid technological advancements, handwritten characters still hold significant roles in various fields, including education [32,33], communication, biometric signature verification [26, 27], and health care [35]. These applications often require

digitization of the handwritten characters and associated hand movements to facilitate effective analysis and interpretation of the underlying task. Offline and online handwriting recognition are crucial steps involving the digitization of handwritten characters [74]. Most existing systems actively use image-processing techniques highly sensitive to environmental lighting conditions. Surface Electromyography signals (sEMG), being invariant to lighting conditions, are used in online handwriting recognition to facilitate the automatic transcription of handwritten characters.

Handwriting recognition has been an important research problem and is broadly divided into online and offline handwriting recognition [75].In offline recognition, the handwritten words or characters are determined from the scanned images using the spatial-luminance properties. On the other hand, online recognition can transduce the handwritten text into electronic data by processing various Spatio-temporal information such as pen tip trajectory (successive two-dimension coordinates), velocity & its magnitude, the pressure exerted on the tip, the number of strokes, etc. [75,76]. In addition to traditional approaches, a few online recognition systems have also used sEMG signals for recognition tasks [14]. These signals are used to model a relationship between muscle activity and handwriting movements.

Handwriting motions are a product of complex cognitive processes involving bioelectric signal stimuli generated through the nervous system. The sEMG signals can capture this information related to neuro-muscular interaction, which can be further used to reconstruct and identify different handwriting motions [77]. Apart from use in handwriting recognition, understanding the relationship between the sEMG signals and the handwriting movements is beneficial as it can be used in various other fields such as biometric authentication [26, 27], signature verification [28,29], Hand gesture recognition [30], e-learning and academically assisting students to write [31] [32, 33]. Also, the movement caused during handwriting a text can be used to diagnose neurological diseases [78] such as Parkinson [34] and Alzheimer [35]. A model capable of accurately classifying such complex hand movements can help to improve understanding of the relationship between sEMG (muscle activation

pattern) and the corresponding hand movements. The model and the methodologies used can be a foundation for developing prosthetic devices with a higher degree of freedom and precision control [3].

In spite of several advantages, little effort has been made to evaluate and analyze the modeling efficiency of sEMG signals corresponding to handwriting movements. However, few papers have considered the problem of decoding handwritten words using sEMG signals. Work done in these papers can be summarized into two major groups. First, propose efficient mathematical models to reconstruct handwritten traces. Second, an efficient algorithm to recognize handwritten characters must be found.

Reconstruction of the handwritten text is mainly used in Pen computing (online handwriting recognition). Pen computing is a field that constitutes computer-based applications where input is generally provided with a pen or a stylus. These applications include a computer-based graphical input interface along with a handwriting recognition mechanism that processes the user's handwriting as input and can identify and interpret the handwritten words or characters. Early work in handwriting reconstruction applied linear mathematical model [14,79] and non-linear analysis [80] to map the relationship between handwriting motion(pen-tip displacement) and the corresponding muscle activity signals. Linderman et al. [14] use the Wiener filters for reconstructing the handwritten characters, achieving maximum reconstruction accuracy of 73% and 49% for Y and X coordinates, respectively. Wheras, Okorokova et al. [81], improved the previous reconstruction accuracy by using Linear Kalman filter. In their work, the reconstruction was performed on the basis of sEMG measurement and the information related to handwriting kinematics. Moreover, Chihi et al. [82] authors proposed a multi-model approach using the Recursive Least Squares algorithm to characterize handwriting. They used two-channel sEMG signals and claimed that using a multi-model structure increases flexibility in modeling different graphic traces written by a specific person. In [83], the authors used the concept of

the internal observer to improve the reconstruction accuracy. They claim that their approach allows for overcoming the parameter learning problem.

The initial model for handwritten character recognition using sEMG signal was proposed in the work of Linderman et al. [14]. The authors used a template matching technique to refine the sEMG segments, followed by applying Fischer linear discriminant analysis to translate the signals to characters. Their proposed approach achieved recognition accuracy of 90% for ten digits (0-9). Later, Huang et al. [84] proposed a Dynamic Time Wrapping(DWT) based method for handwriting recognition, achieving an accuracy of 84.26%. An improvement on the DWT-based handwriting recognition approach was proposed in [85]. The authors used a two-phase template matching approach to extract more relevant features, while the Mahalanobis distance was used instead of Euclidian distance to minimize the inter-class variance. With their modification, the authors claim to achieve an improvement of 9.20%, achieving 92.42% recognition accuracy. Inspired by the success of deep neural networks facilitating automatic feature extraction, authors in [86] applied a hybrid network using a Convolution layer and Recurrent neural network (RNN) for multi-stroke handwritten character recognition. With the dataset consisting of sEMG signals corresponding to 26 English alphabets and ten digits, their model achieved the maximum recognition accuracy of 94.85%. The accuracy achieved in this paper is among the highest recognition accuracies reported for the handwriting character recognition tasks.

Despite these advancements, significant limitations remain in this domain. The existing approaches often necessitate elaborate preprocessing and sophisticated feature extraction methods, which are both resource-intensive and computationally expensive. Simpler methods like template matching can achieve high accuracy in controlled environments, while advanced techniques like deep learning offer robust

performance across more complex datasets, albeit with higher computational demands. Variability in sEMG signal quality due to factors such as electrode displacement and muscle fatigue can also detract from the system's performance. Furthermore, while the classification accuracies achieved by existing methods are notable (approximately 97%), even higher recognition accuracy is essential for developing practical, real-world applications. Ongoing research is required to further enhance the accuracy, reliability, and usability of these recognition systems. Table 2.2 summarizes the related work in this domain.

TABLE 2.2: Summary of the related work for sEMG senors-based handwriting recognition tasks

Paper	Method used	Number of subjects	Number of classes	Sensor used	Recognition Accuracy
[14]	Template matching	6	10	sEMG	97.00%
[84]	Dynamic time wrapping	3	26	sEMG	84.29 %
[85]	Dynamic time wrapping	4	26	SEMG	92.42 %
[86]	Deep learning (LSTM+RNN)	3	36	sEMG	94.85%

2.2.1 A Multi-Modal Sensors Feature Fusion Approach for Handwritten Characters Recognition

Multi-modal deep architecture has been utilized in various applications for improved performance, including medical imaging [87, 88], computer vision for tasks like image captioning, visual question answering [89, 90], and speech recognition for phoneme sequence & audio signal [91, 92]. In natural language processing, the multi-modal deep learning models have been instrumental in speech-to-text translation [93] and audio captioning [94]. It has also found applications in robotics and autonomous vehicles [95, 96], where they facilitate the fusion of sensor data from diverse modalities such as visual, lidar, and radar. These fusion processes have been credited with enhancing the accuracy of perception and decision-making within these systems [97].

Handwritten character recognition can benefit significantly from incorporating multi-modal sensing capabilities. This approach facilitates capturing multiple data

streams, accommodating information including hand and finger movements, user posture, and body motions. These different data streams can be combined to create a more accurate and robust model.

The benefits of using multi-modal wearable sensors include providing a comprehensive and accurate view of human activity and hand gesture recognition compared to using a single sensor or modality. For example, combining complementary sensory modules such as data from an accelerometer and a gyroscope can provide more accurate information about the orientation and movement of a hand, thereby improving the accuracy of hand gesture recognition. Researchers have utilized a range of sensing modalities, including Inertial Measurement Units (IMUs) [98–101], microphones [102, 103], WiFi signals [104, 105], and force sensors [106, 107], and their combinations, in their efforts to develop more accurate and reliable models for handwriting recognition. Handwriting recognition models have been used in different scenarios that involve generating manual handwriting gestures, such as in the air, writing on a whiteboard, and using pen-and-paper-based approaches.

Wu et al. [108] used an accelerometer sensor and developed frame-based descriptors for recognizing 12 different gestures. Their approach achieved recognition accuracy of 89.29% using a multi-class SVM classifier, demonstrating the potential of accelerometer- based techniques.

Building upon this work, Wang et al. [109] introduced a digital pen equipped with an accelerometer, a microcontroller, and an RF wireless transmission module for recognizing handwritten characters. Their model achieved an impressive recognition accuracy of 98% for handwritten digits. These are the initial prominent works showcasing and demonstrating the potential of accelerometer-based techniques. Although it was limited to only ten digits. This development showcased the potential of using accelerometer-based technology for character recognition tasks.

Kim et al. [110] proposed a gyroscope-based handwriting recognition system that optimized algorithm efficiency using DTW with a stepwise lower-bounding

technique. With this approach, they achieved a recognition accuracy of 95% for handwritten characters, showcasing the effectiveness of gyroscope sensors in character recognition. Whereas Xu et al. [111] utilized accelerometer and gyroscope sensor data collected from smartwatches to recognize hand and finger movements. Their system achieved a recognition accuracy of 95% for the 26 alphabets, demonstrating the potential of wearable devices for gesture and character recognition. However, for real-world applications, relatively higher accuracy is desirable.

Deep learning models have also made significant contributions to gesture and character recognition. Li et al. [112] developed a hybrid model that integrated Bi-LSTM and Bi-GRU and leveraged Fisher's criterion to balance intra-class and inter-class variations. Their proposed model achieved an impressive recognition accuracy of 98.04%. However, the evaluation dataset was limited to a set of gestures consisting of six Arabic numerals and six English alphabets.

Wang et al. [113] took a different approach by developing an augmented reality (AR)-based system for simulating a real whiteboard. Their framework employed an AR controller to imitate handwritten characters, achieving a 72% improvement in efficiency compared to traditional methods. Younas et al. [114] focused on reconstructing finger movements using low-cost IMU sensors. Their framework achieved recognition accuracy of 95.01% for uppercase English alphabets, but their dataset was limited to 5270 instances.

Ott et al. [107] utilized a pen equipped with sensors (accelerometer, gyroscope, and force) and applied baseline deep learning models (CNN and LSTM) to recognize different English alphabets. Their models achieved recognition accuracies of 83% for writer-dependent and 90% for writer-independent handwritten characters. [86] used wearable sensors to collect signals generated from muscle activation to map the relation between handwritten character movement and surface Electromyography signals. Their work resulted in 94.58% recognition accuracy, showcasing the potential of these signals for the handwriting recognition task.

In another work, Bu et al. [115] introduced a flexible handwriting recognition model that was less dependent on specific pens or writing platforms. Their model, equipped with IMU sensors, successfully inferred characters written on surfaces such as blackboards and notebooks. By exploiting principal component analysis, they achieved a recognition accuracy of 98.2%. Luo et al. [116] proposed an approach leveraging IMU and DTW methods to recognize characters written in the air. Their framework, incorporating a new optimization mechanism, achieved a recognition accuracy of 84.6% for 26 uppercase alphabets.

In contrast, Hendy et al. [117] introduced a novel framework capable of recognizing characters handwritten in the air using a single ultra-wideband radar (UWB). By employing deep learning models and representations, they achieved impressive recognition accuracy of 98.5% for ten different characters with a total of 1800 samples.

In another work by [116], the authors proposed a framework for real-time recognition of characters written in the air using IMU sensor. By utilizing constrained dynamic time warping and baseline machine learning models, they achieved a classification accuracy of 88.9% for 10 English alphabets. In a recent study, Tripathi et al. [118] proposed an air writing recognition model that identified finger movements corresponding to uppercase alphabets. They recorded signals using sEMG sensors and employed deep learning models for the recognition task, achieving a recognition accuracy of 78.50% using short-time Fourier transform and 2-D convolutional neural networks (CNN).

These studies highlight various multimodal sensor-based handwriting recognition methods, highlighting their accuracy, user comfort, computational demands, and generalizability across diverse handwriting tasks. A review of the current literature reveals a limited exploration of the potential of combining sEMG and IMU signals for handwriting recognition. Table 2.3 summarizes the related work in this area.

TABLE 2.3: Summary of the related work for multimodal senors-based handwriting recognition tasks

References	Technique used	No of gestures	Sensors used	Device used for data collection	Gestures Modularity	Performance Metrics (Accuracy)
[111]	Feature extraction with ML classifiers	26 alphabets	Accelerometer and gyroscope	Smartwatch	3D gestures made with finger	95%
[112]	Deep learning (BLSTM and BGRU)	12 gestures (digits+ alphabets)	Accelerometer and gyroscope	Smartphone	3D gestures	99.15% (alphabets) 99.36% (digits)
[114]	Deep learning 1D-CNN	Digits(0-9) Alphabets(a-z)	Acelerometer+ Gyroscope+ Magnetometer	Fingerworn sensors	In air	95.91%(digits) 93.04% (alphabets)
[107]	Deep learning (CNN, LSTM)	Uppercase alphabets	Acelerometer+ Gyroscope+ Force sensor + Magnetometer	Sensors enhanced pen	On paper	90%
[115]	3D tip estimation + Principal component analysis	Handwritten gestures	Acelerometer+ Gyroscope+ Magnetometer	Pen	Regular writing plains	98.2%
[116]	Dynamic Time Wrapping (DTW)	26 alphabets	Acelerometer+ Gyroscope+ Magnetometer	Fingerworn sensors	In air	84.6%
[117]	Deep learning CNN	Digits (0-9)	UWB	Xethra X4-103	In air	98.5%
[119]	Constraint DTW +K-NN	Digits (0-9)	Acelerometer+ Gyroscope+ Magnetometer	Handworn sensor	In air	88.9%
[120]	DTW	Ten gestures	Ultrasonic positioning +IMU	Pen	Unity 3D platform	99.5%

Developing an application capable of digitizing handwritten characters in real time necessitates a model that achieves high recognition accuracy across a wide range of characters. The recognition models reviewed in the survey are generally constrained either by their accuracy or by the number of characters they can reliably recognize, indicating scope for improvement.

2.2.1.1 Handwriting dynamics as a diagnostic tool for neurological diseases

Handwriting analysis has gained significant importance in the research of neurodegenerative diseases [121]. With its recognized potential, an increasing number of researchers are investigating handwriting as a diagnostic instrument for disorders such as Alzheimer's and Parkinson's disease (PD), underscoring its importance in early detection and monitoring of these conditions.

This section offers a review of the current literature, aiming to summarize the research articles that propose handwriting dynamics as a diagnostic tool for neurological diseases, specifically Alzheimer's and Parkinson's (PD).

Loconsole et al. [25] introduced an approach for classifying Parkinson's Disease (PD) using static and dynamic handwriting features. They utilized five different ANN models optimized with Genetic Algorithms, achieving accuracy rates ranging from 75.75% to 86.15%. Gupta et al. [122] developed age and gender-specific machine learning classifiers to predict Parkinson's disease; Their model achieved an accuracy of 83.75% while classifying females.

Diaz et al. [123] presented a classification model incorporating 1D convolution and Bi-GRUs, achieving top-tier accuracy scores of 93.75% and 94.44% on the PaHaW and NewHandPD datasets related to the Parkinson dataset. Pereira et al. [124] converted 1D signals from smart pens into 2D images, then used CNNs for PD classification. Their innovative majority voting-based schema amalgamation of different CNN classifier outputs achieved an impressive 93.50% accuracy. Taleb et al. [125] similarly converted 1D time series to 2D images, proposing CNN and CNN-BLSTM models for PD detection. Their data augmentation approach boosted accuracy rates from 83.33% to 97.62%.

For Alzihimer, Cilia et al. [126] introduced a unique approach to classifying AD from MRI images, extracting handcrafted features and CNN features from participant-created color and binary images. Their findings emphasized CNN features superiority with 74.65% accuracy. Cilia et al. [126] later expanded on their previous work, utilizing multi-channel TIFF images and incorporating transfer learning. Carfora et al. [127]) experimented with Archimedes Spiral images, by embedding dynamic parameters and leveraging AlexNet's feature extraction capabilities. Their findings reported a 7% accuracy improvement using hybrid images. Meng et al. [128] applied 2D discrete Fourier transform on Archimedes spiral and labyrinth

lattice images.Employing the Decision Tree algorithm and achieving a commendable mean AUC of 0.94.

The above-mentioned works for AD classification are based on image-based analysis. One of the limitations of image-based approaches is that they are computationally expensive. The accuracy of these methods heavily depends on the quality of the input images. Variations in image quality, resolution, and consistency across different imaging devices can introduce noise and artifacts that affect the performance of the algorithms.

In contrast, Cilia et al. [121], proposed a sensor-based approach to classify Alzheimer. The advantage of the sensor-based approaches is that they are less susceptible to lighting conditions and are computationally less expensive. In their work, authors underscored the significance of handwriting dynamics and introduced the DARWIN dataset, featuring handwriting samples from Alzheimer's patients and a control group (Cilia et al. [129]). Mwamsojo et al. [130], in their recent work, presented a comparison between RNN, BiLSTM, and CNN methods. Factoring in energy costs alongside accuracy, they recorded an RNN accuracy of 85%, slightly trailing BiLSTM.

The literature reviewed highlights the necessity for developing more accurate models. Furthermore, there has been limited exploration into the application of explainable AI (XAI) [131], which could enhance transparency in the models' decision-making processes, thus building trust and promoting clinical adoption. XAI has the potential to explain why certain features are predictive, enabling clinicians to make more informed decisions in complex and sensitive fields such as medical diagnostics.Using the DARWIN dataset [121], we explored the application of Explainable AI (XAI) techniques for Alzheimer's prediction. Our goal was to create models that not only achieve high predictive accuracy but also provide interpretable and transparent insights into the decision-making process.

2.3 EMG-Based Prosthetic Hand Grasp Recognition

Electromyography (EMG) has been used for prosthetic operations since 1948, with the first clinical myoelectric prosthetics appearing in the 1960s [132,133]. Over the years, various EMG-based control schemes have been devised to mimic natural hand movements in prosthetic hands [134]. The traditional approach uses an off/on approach where a single or two groups of muscles are responsible for closing or opening hand movements [135,136]. This approach uses a suitable threshold to deal with noise and provides limited functionality with sequential control. Later, proportional control was proposed where, in addition to basic opening and closing movements, the user could manipulate one of the mechanical outputs, such as velocity or force [137]. However, the proposed proportional control still lacks the ability to produce finer movements. Direct control was suggested to deal with the limitation of proportional control in which an individual sEMG signal was assigned to a function. But the major issue with Direct control was that their sensitivity was affected by the crosstalk of the sEMG signals [138]. Later, EMG-based pattern recognition (PR) algorithms were deployed to provide multifunctional control by classifying various muscle activation patterns [139]. Despite being used in various applications, the sEMG signals are challenging to analyze and process. Their characteristics are easily influenced by variables such as muscle fatigue, electrode placement, muscle contraction intensity, and limb orientation [140–145].

Analyzing and processing sEMG signals remain challenging due to factors like muscle fatigue, electrode placement, contraction intensity, and limb orientation [140–145]. Machine learning algorithms can address these challenges, enabling the development of models that accurately recognize patterns in sEMG signals for PR-based control, thereby improving the reliability and functionality of hand gesture recognition in prosthetic devices [146].

Research utilizing sEMG signals for hand movement classification dates back to the early 1970s. Over the years, pattern recognition-based (PR-based) sEMG control strategies have gained significant importance, with machine learning algorithms enhancing their efficiency. Various algorithms, such as Linear Discriminant Analysis, log-linearized Gaussian mixture networks, and k-nearest neighbor, have been applied to classification tasks [19, 147].

Notable research in this area includes the work by Spanis et al. [148], where Empirical Mode Decomposition was used for feature extraction, achieving 86.64% accuracy for six hand grasps with a linear classifier. Ruangpaisarn et al. [149] employed single value decomposition and Sequential Minimal Optimization to train an SVM for sEMG-based hand grasp classification. Ouyang et al. [150] utilized sEMG dynamic properties with recurrence quantification analysis to classify four hand grasps, validated by a neuro-fuzzy system. Akben et al. [151] used a cascaded classifier for five basic hand gestures with two-channel sEMG signals, achieving 94.72% accuracy. Coskun et al. [152] applied a deep learning model (1D CNN) for six-hand grasp classification, achieving 94.94% accuracy on a benchmark dataset.

While reviewing the literature, we were inspired by Coskun et al.'s methodology, which reported 94.94% classification accuracy using single-channel sEMG signals on a benchmark dataset [153]. Focusing on a minimal number of channels is beneficial for developing cost-effective robotic and prosthetic hands [154]. The reported accuracy using a single channel motivates us to further improve state-of-the-art accuracy.

Moreover, deep learning has recently achieved state-of-the-art accuracy for sEMG-based classification tasks, though it provides limited insight into the relevant features. This limitation presents an opportunity to apply explainable AI (XAI) to bridge this gap. Table 2.4 summarizes the related work in this area.

TABLE 2.4: Summary of recent methods focused on the minimal number of sEMG channels for classifying different hand grasp movements.

Article	Dataset	sEMG channels	Gestures	Methodology	Accuracy
[148]	[153]	2	Basic hand grasp	Linear classifier on feature extracted using Intrinsic Mode Functions.	86.64%
[155]	[153]	2	Basic hand grasp	Decision tree-based ensemble	71.30%
[156]	[153]	2	Basic hand grasp	K-nn classifier with SVD.& PCA	86.71 %
[157]	[153]	2	Basic hand grasp	K-nn classifier with TQWT.	98.55%
[158]	[153]	2	Basic hand grasp	SVM classifier with Zero crossing.	85.66%
[159]	[153]	2	Basic hand grasp	CNN and reinforcement learning	83.5%
[160]	[153]	2	Basic hand grasp	Bidirecional LSTM architecture	99.12%
[161]	[153]	2	Basic hand grasp	tunable Q factor wavelet transform (TQWT) and shallow classifiers	98.89%
[152]	[153]	1	Basic hand grasp	1-D CNN for classification and feature extraction.	94.94%

2.3.1 Other insightful observations

2.3.1.1 Data segmentation

Effective data segmentation is crucial for accurate hand gesture recognition using surface electromyography (sEMG) data. The reviewed literature identifies two primary methods for segmenting sEMG data: sliding windowing and gesture detection method [19].

Sliding windowing partitions the EMG into consistent segments, which can be either fixed adjacently or fixed overlappingly. Overlapping segments capture detailed and continuous data by sharing portions with previous segments, improving recognition accuracy through smoother transitions, and reducing missed data points.

Gesture detection identifies the start and end of a hand gesture, resulting in variable segment lengths based on gesture duration. This method precisely pinpoints the duration of each gesture, ensuring only relevant data is analyzed.

Both sliding windowing and gesture detection are essential for reliable sEMG-based hand gesture recognition. Sliding windowing offers consistent, detailed segments, especially with overlapping windows, while gesture detection ensures accurate

identification of gesture durations.

2.3.1.2 Response Time of Hand Gesture Recognition Models

To measure the effectiveness of hand gesture recognition (HGR) models, we introduced the variable Response Time. Response Time includes the total duration for data collection (window length) and data analysis for a single instance by a proposed model [162]. For real-time processing, it is crucial to minimize both durations while ensuring sufficient data for accurate hand gesture recognition. For example, in prosthesis control, the optimal data collection time using four sensors with a 1 kHz sampling rate ranges between 150-250 milliseconds.

An HGR model is considered suitable for real-time use if the response time is less than the optimal delay. Reported optimal delays vary, including 100 milliseconds for fast prosthetics, 125 milliseconds for slower ones, and up to 500 milliseconds in other cases [19, 163].

Most of the reviewed HGR models meet the criteria for real-time performance. However, some studies did not report their data collection and analysis times, highlighting a gap in the reporting standards that could affect the assessment of real-time capabilities.

2.3.1.3 Data acquisition device

During the literature review, it was found that electromyograms (EMGs) are collected from various EMG sensors, which can be either homemade or commercial devices. Different hand gesture recognition (HGR) models utilize varying numbers of sensors and sampling rates to optimize data acquisition.

Most of the reviewed HGR models employ eight sensors. Many of these models use commercial devices like the Myo armband, which features eight sensors operating at a sampling rate of 200 Hz. Others use homemade devices with higher sampling rates, ranging from 960 Hz to 1200 Hz, to improve data fidelity. Many models adopt

a sampling rate of 1000 Hz to effectively capture the majority of the signal power, typically below 400–500 Hz.

In addition to the Myo armband, various commercial devices are frequently used, including the MA300, Bio Radio 150, ME6000, Analog Front End (ADS1298), Telemyo 2400T G2, and EMG-USB2. Furthermore, some models employ high-density EMG sensors to enhance data acquisition quality, providing finer spatial resolution and more detailed signal capture.

2.3.2 Preliminaries

2.3.2.1 Myo armband: wearable sensor used for data collection

The Myo Armband, a cost-effective wearable sensor developed by Thalmic Labs, was used to collect sensory data [164]. This device is designed to capture and interpret the electrical activity generated by muscle movements using electromyography (EMG) sensors. Its ability to convert these signals into actionable data makes it a versatile tool across various fields, including human-computer interaction, rehabilitation, and research.

During our experiments, we focus on its sEMG sensor signals for recognition purposes. This wearable device features a medical-grade eight-channel wireless sEMG sensor with a 200Hz sampling rate [165]. The device's processing is managed by a Freescale Kinetis ARM Cortex M4 120Mhz MK22FN1M MCU processor, while data transmission is facilitated through a Bluetooth connection using the BLE NRF51822 chip [164].

The 9-axis inertial measurement unit (IMU) comprises a three-axis accelerometer, a three-axis magnetometer, and a three-axis gyroscope sensor, which collectively gather hand motion-related information. The IMU unit is based on the InvenSense MPU-9150 [164], a low-cost, low-power module commonly used for nine-axis motion tracking.

The Myo armbands facilitate the acquisition of data by transmitting raw EMG signals from the sensors to associated devices. These signals can be processed by software applications for visualization purposes or preserved as Comma-Separated Value (CSV) files. Figure 2.1 illustrates the Myo armband, while Figure 2.2 depicts a subject adorned with a Myo armband during the data collection.

FIGURE 2.1: Myo Armband

FIGURE 2.2: A subject wearing Myo Armband during our dataset collection

.

2.3.3 Datasets

This thesis used different sEMG signal datasets related to static and dynamic gestures to validate the proposed Hand gesture recognition models. A few datasets corresponding to static and dynamic gestures were collected for the experiments.

Datasets used for Static hand gestures recognition

2.3.4 Dataset-I:Ninapro reference dataset(Publicly available)

The Ninapro DB5 dataset has been used as a reference dataset to evaluate the performance of classification algorithms applied over sEMG signals [1] [166]. This dataset consists of sEMG signals collected from 10 healthy subjects over 53 different

movements, including a neutral hand position. These 53 movements were divided into three exercises, collected using Myo Armband. We compared our results with exercise A of this dataset, which consists of 12 different gestures, including finger movements.

The dataset is collected by using two Myo Armbands containing 16 sEMG channels.

2.3.5 Dataset-II: Hand grasp dataset(Publicly available)

For experiments, we utilized the publicly available database provided by Sapsanis et al. [153], which consists of sEMG signals generated for six basic hand movements/-gasps. The sEMG signals were collected by applying two surfaces EMG electrodes on the muscles named Longus & Brevis and Flexor Capri Ulnaris Extensor Capri Radialis. The reference electrode was adjusted in the middle to record the muscle activation correctly. The dataset consists of two sub-datasets, each containing two channel sEMG signals corresponding to these two electrodes.The dataset-I consists of sEMG signals collected from five healthy subjects, including two males and three females aged between 20 and 22 years. The subjects were instructed to repeat the six-hand grasp movements wearing the sEMG electrode. The six-hand grasp movements include Spherical (SL), Tip (TP), Palmar (PR), Lateral (LL), Cylindrical (CL), and Hook (HK) movements depending on the various hand shapes and objects used for the grasp. For a subject, each of the six grasps was repeated 30 times, performing every grasp for a time duration of six seconds. In contrast, dataset II consists of sEMG signals collected from a single healthy subject. The sEMG signals were collected for three consecutive days for a similar number of grasps.

Both datasets were collected with a sampling rate of 500 Hz. The sample length of the signals of dataset-I was 3000, while that of dataset-II was 2500. Figure 2.3 shows the raw sEMG signals generated corresponding to the six-hand grasp movements.

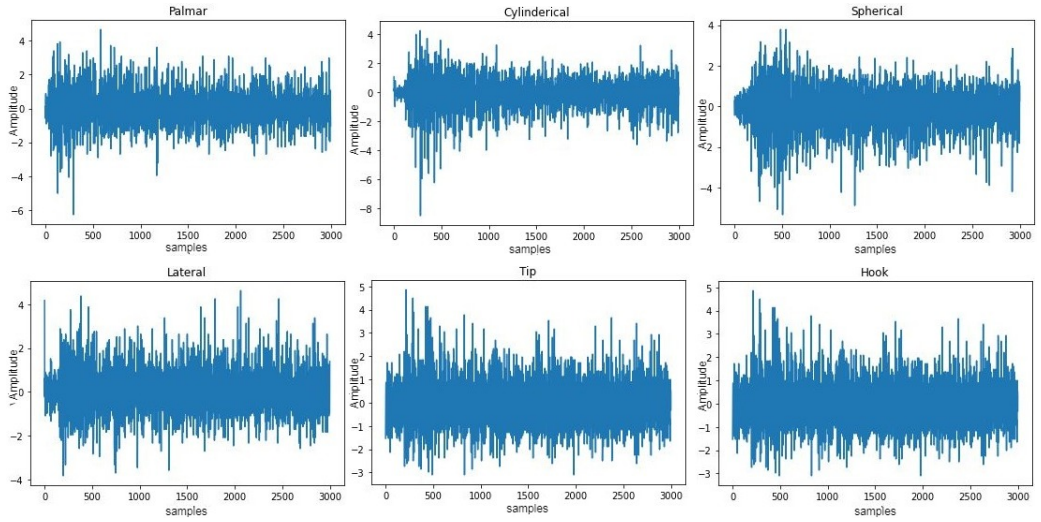

FIGURE 2.3: Raw sEMG signals corresponding to six hand grasp movements used in the dataset

2.3.6 Dataset-III: American Sign Language digit dataset, ASL-10 (Collected Dataset)

The dataset consists of sEMG signals collected from 20 healthy subjects aged 22-28 years, comprising 15 males and five females. Each of the subjects was made aware of American Sign Language(ASL) digit gestures through videos and proper illustrations. Using the data acquisition and sensor placement protocol, each of the subjects is made to generate a gesture for each of the ten ASL digits. For the ASL-10 dataset, the data was collected and processed for only two sEMG sensors. Though the sampling frequency for the device was lower, a total of 53000 samples were collected and analyzed. For two sEMG signals, around 900 features were extracted. Figure 2.4 shows the different static hand gestures for digits 0 to 9 in ASL, while the raw sEMG signals for ASL-10 are shown in Figure 2.5

2.3.7 Dataset-IV: ASL manual alphabet dataset, ASL-24 (Collected Dataset)

The dataset includes 24 ASL manual alphabets used by new ASL signers in fingerspelling words, where each alphabet is formed by different hand shapes. Out

FIGURE 2.4: Different hand gestures for digits 0 to 9 in ASL-10.

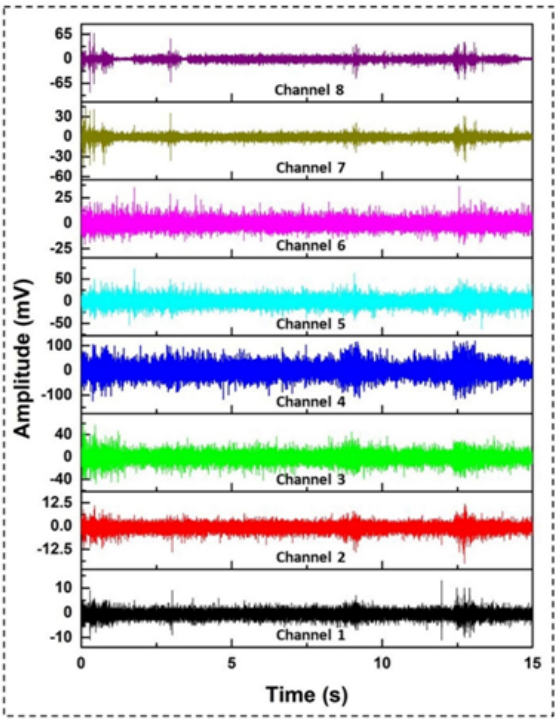

FIGURE 2.5: Raw sEMG signals patterns of a subject for digit 9 of ASL

of the 26 ASL manual alphabets corresponding to the letters A-Z, we included 24 static signs, excluding 'j' and 'z' due to their dynamic hand movements.

Data was collected from 10 subjects: eight men and two women aged 22-28 years. The experimental protocol, data acquisition, and dataset preparation steps

were designed to be consistent with the ASL-10 dataset. Eight sEMG sensors were used to capture signals for the 24 ASL manual alphabets to accommodate the increased number of gestures. These sensors were placed on the right hand of each subject, with two sensors positioned at the Extensor Carpi Ulnaris and Extensor Carpi Radialis Longus muscles.

In total, 83,000 samples for the 24 ASL gestures were collected and processed. From the eight sEMG signals, approximately 3,600 features were extracted. Figure 2.6 highlights the different ASL manual alphabets used in the ASL-24 dataset.

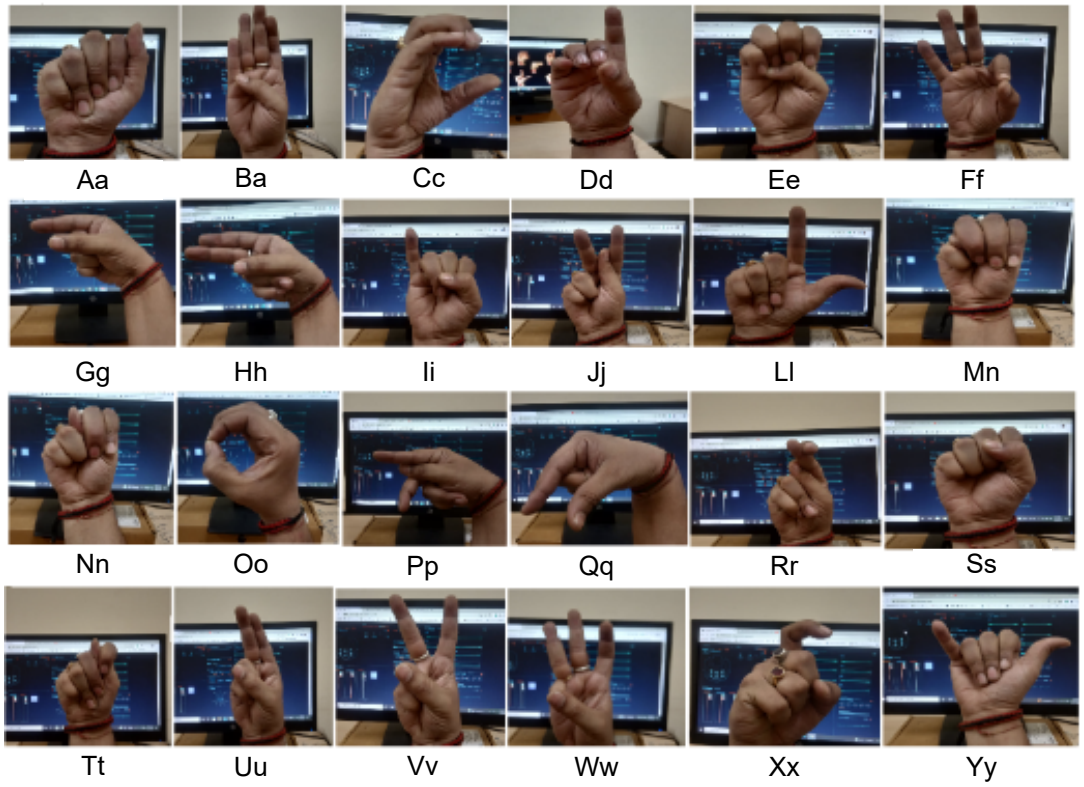

FIGURE 2.6: Figure illustrating the different ASL manual alphabets used in ASL-24 dataset

Datasets used for Dynamic hand gestures recognition

2.3.8 Dataset-V: sEMG based Handwritten character recognition dataset, S-HCR (Collected dataset)

For the S-HCR dataset, sEMG signals corresponding to 26 English alphabets were collected from various subjects. While each subject wrote each alphabet using pen and paper, sEMG signals from eight sensors were recorded and stored in CSV format. The data collection involved fifteen healthy participants (eleven males and four females) aged between 28 and 32 years. A wearable sensor from Thalmic Labs, equipped with a wireless eight-channel sEMG and an IMU sensor, was used to gather the signals, operating at a sampling frequency of 200Hz for the sEMG sensors. The raw signals from each subject's sensors were stored and processed to create different datasets. Each subject was instructed to write each alphabet 375 times, resulting in a total of 146,250 samples. Figure 2.7 shows the template used template corresponding to 26 lower cases of the English alphabet used for collected raw signals.While the Figure 2.8 depicts the Experimental setup used during data collection.

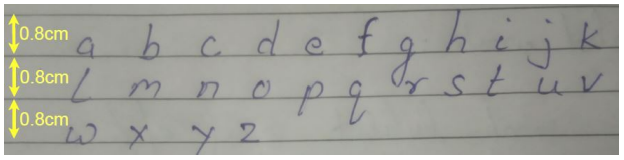

FIGURE 2.7: The figure illustrating the template corresponding to 26 lower cases English alphabet used for collected raw signals

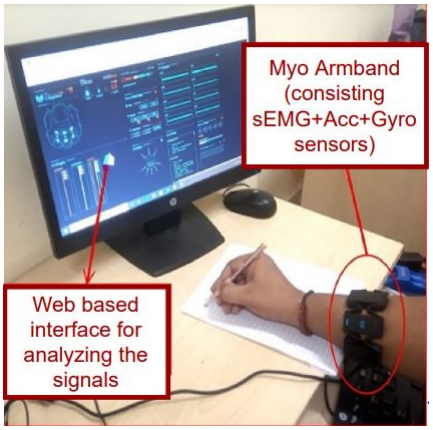

FIGURE 2.8: Experimental setup used during data collection

2.3.9 Dataset-VI: Multi-Modal Handwritten Character Recognition dataset, MM-HCR (collected dataset)

We collected a Multi-Modal Handwritten Character Recognition (MM-HCR) dataset, which includes data from sEMG (surface electromyography) and IMU (inertial measurement unit) sensors for 26 lowercase handwritten English alphabets written on a whiteboard. The sEMG and IMU readings were recorded and saved in CSV format.

To minimize the influence of muscle fatigue on EMG signals, we followed a protocol as described in [167]. Data was collected over multiple sessions, each consisting of several sets where participants repeated 10-15 characters. A 15-second interval was maintained between sets. In total, approximately 21,000 samples were gathered for the 26 alphabets.

To standardize the size of the characters, two parallel lines spaced 6 cm apart were drawn on the whiteboard. Participants were provided with a template displaying all 26 lowercase alphabets to ensure consistency in their writing. Figure 2.9 shows a participant writing with the MyoArmband during the experiment, while Figure 2.10 shows the template used while collecting dataset.

The IMU data captured hand orientation and movement during the writing process, including readings from accelerometers, gyroscopes, and magnetometers.

2.3.10 Dataset-VII: DARWIN, used for Alzheimer prediction (Publicly available)

We utilized the DARWIN dataset [168], a benchmark resource that collects various handwriting dynamics data, which can be used to detect Alzheimer's disease.

Handwriting data were collected following a protocol based on Cilia et al. [121], which involved 25 tasks. These tasks included Graphic (G), Copy (C), Memory (M), and Dictation (D) activities designed to evaluate different aspects of a participant's

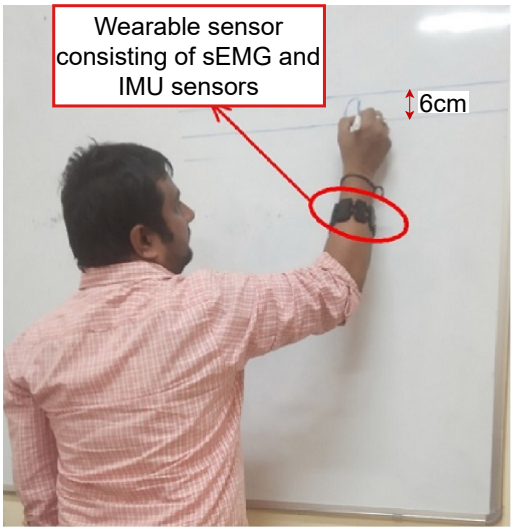

FIGURE 2.9: A participant writing with the MyoArmband on the whiteboard.

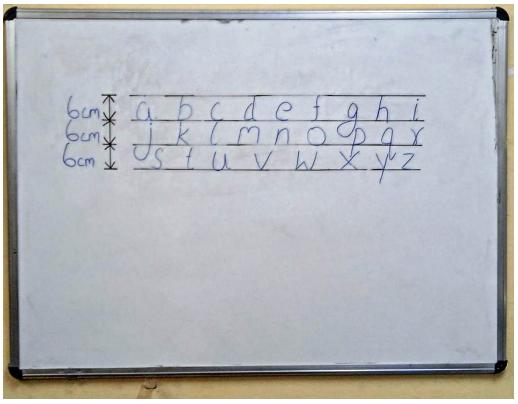

FIGURE 2.10: The 26 lowercase English template used during data collection.

writing abilities. The dataset comprises data from 174 participants, including 89 Alzheimer's disease (AD) patients and 85 healthy individuals. Participants were recruited based on cognitive tests such as MMSE, FAB, and MoCA, excluding those on psychotropic drugs or with compromised cognitive abilities. Demographic factors like age, education, job type, and gender were matched among participants.

Data acquisition was performed using a Wacom Bamboo tablet with a pen, capturing x-y coordinates of pen movements at 200Hz and differentiating between "on-paper" and "in-air" movements. During data collection, participants used a set of 25 paper sheets with task descriptions. The tablet, connected to a PC, displayed real-time writing movements. After completing the tasks, the raw data were processed to extract relevant features.

From the raw data, 18 features were extracted, including total time spent on a task, time for in-air movements, average speed, and acceleration on paper. Tables 2.5 and 2.6 provide details on the features extracted and the tasks performed. A total of 25 tasks were conducted to compile the DARWIN dataset.

TABLE 2.5: List of Tasks Performed and Their Categories

Task	Description and Category	Task	Description and Category
1	Draw a signature (M)	14	Memorize and rewrite: 'telefono', 'cane', 'negozio' (M)
2	Connect points with horizontal lines four times (G)	15	Reverse-copy the word 'bottiglia' (C)
3	Connect points with vertical lines four times (G)	16	Reverse-copy the word 'casa' (C)
4	Retrace 6 cm diameter circle four times (G)	17	Copy six distinct words in boxes (C)
5	Retrace 3 cm diameter circle four times (G)	18	Write the object's name from a picture: chair (M)
6	Copy letters 'l', 'm', 'p' (C)	19	Copy postal order fields (C)
7	Copy letters from nearby rows (C)	20	Dictate and write a simple sentence (M)
8	Cursively write 'l' four times smoothly (C)	21	Retrace a complex shape (G)
9	Cursively write bigram 'le' smoothly (C)	22	Copy a phone number (C)
10	Copy the word 'foglio' (C)	23	Write a dictated phone number (M)
11	Copy 'foglio' over a line (C)	24	Draw an 11:05 clock (G)
12	Copy the word 'mamma' (C)	25	Copy a text paragraph (C)
13	Copy 'mamma' over a line (C)		

TABLE 2.6: Features and their Descriptions

Metric	Description	Metric	Description
TT	Duration for the entire task.	AT	Duration for in-air movements.
PT	Duration for on-paper movements.	MSP	Average speed for on-paper movements.
MSA	Average speed for in-air movements.	MAP	Average acceleration for on-paper movements.
MAA	Average acceleration for in-air movements.	MJP	Average jerk for on-paper movements.
MJA	Average jerk for in-air movements.	PM	Average pressure exerted by the pen tip.
PV	Variance in pen tip pressure.	GMRTP	Generalized Mean Relative Tremor for on-paper movements.
GMRTA	Similar metrics for in-air movements.	GMRT	Average of GMRTP and GMRTA.
PWN	Total pendowns during the task.	XE	Farthest difference along the X-axis.
YE	Same metric but for the Y-axis.	DI	Measures handwritten traces spread on paper.

2.3.11 Dataset-VIII: UCI datasets: For validating proposed metaheuristic feature section approach

A diverse range of datasets from various fields was carefully selected to ensure the generalizability of the outcomes. These datasets vary in their attributes, the scale of their feature sets, and the types of classification classes, including both binary and multi-class categories. This diverse selection enables a comprehensive evaluation across different data types and complexities, providing a robust test of the methodology's effectiveness in varied scenarios. The details of these datasets are provided in Table 2.7.

TABLE 2.7: Publicly available dataset used for the experiments

Dataset	Attributes	Classes	Instances
Chess [169]	36	2	3196
Tic-tac-toe [170]	9	2	958
Libras [171]	91	15	360
CNAE [172]	857	9	1080
Ionosphere [173]	34	2	351
Car [174]	6	4	1728
Wine [175]	13	3	178
German [176]	20	2	1000
Zoo [177]	16	7	101
Promoter [178]	58	2	106
Parkinsons [179]	22	2	197
Sonar [180]	60	2	208
Splice [181]	61	3	3190
Turkish [182]	50	3	400

2.3.12 Evaluation Metrics

We assessed the performance of the proposed machine learning pipeline to determine its effectiveness as a recognition module within a sign language recognition system. Key factors impacting the efficiency of such a module include recognition time, accuracy, pervasiveness, scalability, and invasiveness, as noted by Paudyal et al. [69]. In this study, we concentrated on evaluating recognition time, accuracy, and system scalability.

To measure recognition accuracy, we framed the ASL gesture recognition task as a multi-class classification problem. In addition to accuracy, we evaluated other widely accepted metrics for multi-class classification [183]. These metrics include the Receiver Operating Characteristic (ROC) curve, Kappa score, and Matthews correlation coefficient (MCC), defined as follows:

2.3.12.1 Accuracy

is calculated using the formula:

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.1}$$

Accuracy measures how correctly the classification model predicts the given dataset.

2.3.12.2 Matthews Correlation Coefficient (MCC)

The Matthews Correlation Coefficient (MCC) is a more reliable metric for classification performance, especially in the presence of imbalanced datasets. For binary classification, MCC is defined as:

$$MCC_{bi} = \frac{tp_{(bi)} \cdot tn_{(bi)} - fp_{(bi)} \cdot fn_{(bi)}}{\sqrt{(tp_{(bi)} + fp_{(bi)})(tp_{(bi)} + fn_{(bi)})(tn_{(bi)} + fp_{(bi)})(tn_{(bi)} + fn_{(bi)})}} \tag{2.2}$$

Here, $tp_{(bi)}$, $tn_{(bi)}$, $fp_{(bi)}$, and $fn_{(bi)}$ refer to true positives, true negatives, false positives, and false negatives in a binary classification context. For multi-class classification, MCC is extended using the confusion matrix C as:

$$MCC_{mc} = \frac{f \cdot e - \sum_n^N p_n \cdot t_n}{\sqrt{(e^2 - \sum_n^N p_n^2)(e^2 - \sum_n^N t_n^2)}} \tag{2.3}$$

where C is the confusion matrix of size $n \times n$, $t_n = \sum_p^N C_{pn}$ denotes the number of times class n correctly occurred, $p_n = \sum_p^N C_{pn}$ is the number of times class n was predicted, $f = \sum_n^N C_{nn}$ denotes the total number of correctly predicted instances, and $e = \sum_p^N \sum_q^N C_{pq}$ represents the total number of samples.

2.3.12.3 Kappa Score

The Kappa score measures the agreement between the actual and predicted classes by a classifier, accounting for chance agreement. It is computed using the confusion matrix C as:

$$Kappa_{mc} = \frac{f \cdot e - \sum_n^N p_n \cdot t_n}{e^2 - \sum_n^N p_n \cdot t_n} \tag{2.4}$$

This metric is particularly useful for evaluating classification performance in scenarios where the classes are imbalanced.

2.3.12.4 Receiver Operating Characteristic (ROC) Curve

It evaluates the model's ability to differentiate between classes. In multiclass settings, the ROC curve and AUC are expanded from their original binary format to accommodate multiple classes by treating each class as a separate binary classification (one vs. rest). This allows for evaluating how well the model distinguishes each class from others. The overall performance can be summarized using either a macro-average or a weighted average of the AUC values from each binary comparison. This method provides a comprehensive view of the model's discriminative power across multiple classes.

2.3.12.5 Response Time

To analyze the computational overhead of a real-time recognition system built using our proposed pipelines, we tried to approximate end to end processing delay of such a system. The end-to-end delay can be approximated by calculating the individual delay of all the major components of an online classification system. Mainly, an online classification system consists of various modules, including data acquisition (buffer storage)& segmentation (windowing), feature extraction, and class label prediction using a pre-trained model [73]. We quantified the summation of each component's time delay in a single term called response time. The total delay, referred to as Response Time (T_{Res}), reflects the time the trained classifier takes to predict a single gesture. Let the window size used in the model be denoted as $W_s(s, p)$. The Response Time is calculated as:

$$T_{Res} = T_{w_s(s,p)} + T_{fe} + T_{Recog} \tag{2.5}$$

Where:

- $T_{w_s(s,p)}$ is the time to extract a window segment $W_s(s, p)$ from the raw input signal,

- T_{fe} is the time for feature extraction from the window segment,
- T_{Recog} is the time to predict the label for the window segment.

A detailed analysis of response time will be referred to throughout various chapters of this thesis, as it is a critical metric for evaluating the performance and efficiency of the proposed system.

2.3.12.6 Explainable AI (XAI): Leveraging SHAP (SHapley Additive exPlanations) for Interpreting Machine Learning Models

Explainable AI (XAI) is an essential area of artificial intelligence focused on making machine learning models transparent and comprehensible to humans [131]. One of the key techniques in XAI is SHAP (SHapley Additive exPlanations), which utilizes concepts from cooperative game theory, specifically the Shapley value, to assign each feature in a model's input a value that reflects its contribution to the model's prediction [184]. The Shapley value is a method to distribute payouts fairly among players, and in the context of machine learning, it ensures that the contribution of each feature is fairly assessed [185].

By leveraging SHAP, one can decompose a prediction into the sum of the feature contributions and the model's baseline output (i.e., the expected output if no features are used). SHAP values provide both local and global interpretability: they explain individual predictions by showing how each feature impacts the specific instance and offer insights into the overall model behavior by aggregating these local explanations [184, 186].

Implementing SHAP involves calculating each feature's marginal contribution across all possible feature combinations. While theoretically robust, this computation can be computationally expensive for models with many features. To mitigate this, various approximations and efficient algorithms like Kernel SHAP [184], Tree SHAP [187], and Deep SHAP [184] have been developed, tailored to different types

of models (e.g., kernel-based models, tree-based models, and deep neural networks, respectively).

One key advantage of SHAP is its ability to handle complex models, including black-box models, using a unified framework. It provides model-agnostic explanations, meaning it can be applied to any model without requiring knowledge of its internal architecture [186, 188]. This flexibility enables SHAP to be used across various domains and machine-learning techniques.

SHAP helps visualize and interpret the importance of features and interaction effects and detect biases in the model. Tools and visualizations such as SHAP summary plots, dependence plots, and force plots enable a detailed and intuitive understanding of model behavior, which is crucial for building trust in AI systems, enabling better debugging, and ensuring that models make fair and justifiable decisions [184].

In summary, SHAP follows additive feature attribution methods by decomposing a model's prediction into contributions from individual features using Shapley values. It considers all possible combinations of features and calculates the average marginal contribution of each feature, ensuring that the contributions are additive and providing interpretable explanations for model predictions. Details and theoretical background are discussed in Section 4.3.1.

2.4 Feature Selection Algorithms

This section explains the fundamentals of four essential feature selection algorithms used in the thesis for the proposed ensemble feature selection in Chapter 3: ReliefF, ANOVA, Chi-square, and Mutual Information. These algorithms help identify the most relevant features for classification tasks by evaluating their importance based on various statistical principles.

2.4.1 ReliefF Algorithm

The ReliefF algorithm [189], an extension of the original Relief algorithm [190], is designed to handle multi-class problems, improve robustness, and deal with incomplete and noisy data. It estimates the quality of attributes according to how well their values distinguish between instances that are near each other. The algorithm updates the quality estimation for each attribute by considering the differences between instances of the same class (hits) and instances of different classes (misses), adjusting weights accordingly. ReliefF handles incomplete data by treating missing values probabilistically, ensuring that the algorithm remains effective even with imperfect data. The probabilistic interpretation of ReliefF's quality estimation provides a robust measure of an attribute's relevance, making it a powerful tool for feature selection. Algorithm 1 outline the working of ReliefF.

Algorithm 1 ReliefF Algorithm

Input: for each training instance, a vector of attribute values and the class value

Output: the vector W of estimations of the qualities of attributes

set all weights $W[A] := 0.0$
for $i := 1$ to m **do**
    randomly select an instance R_i
    find k nearest hits H_j
    for each class $C \neq class(R_i)$ **do**
        from class C find k nearest misses $M_j(C)$
    end for
    for each attribute A **do**
        $W[A] := W[A] - \sum_{j=1}^{k} \frac{diff(A,R_i,H_j)}{m \cdot k}$
        $W[A] := W[A] + \sum_{C \neq class(R_i)} \left[\frac{P(C)}{1-P(class(R_i))} \sum_{j=1}^{k} \frac{diff(A,R_i,M_j(C))}{m \cdot k} \right]$
    end for
end for

2.4.1.1 Equations

The quality estimation for an attribute A is updated as follows:

$$W[A] := W[A] - \sum_{j=1}^{k} \frac{diff(A, R_i, H_j)}{m \cdot k} + \sum_{C \neq class(R_i)} \left[\frac{P(C)}{1 - P(class(R_i))} \sum_{j=1}^{k} \frac{diff(A, R_i, M_j(C))}{m \cdot k} \right] \tag{2.6}$$

2.4.1.2 Handling Incomplete Data

For handling incomplete data, the difference function $diff$ is modified. Missing values are treated probabilistically:

$$diff(A, I_1, I_2) = \begin{cases} 1 - P(value(A, I_2)|class(I_1)) & \text{if } I_1 \text{ has missing value} \\ 1 - \sum_V (P(V|class(I_1)) \cdot P(V|class(I_2))) & \text{if both have missing values} \end{cases} \tag{2.7}$$

2.4.1.3 Probabilistic Interpretation

ReliefF's estimate $W[A]$ of the quality of attribute A can be interpreted probabilistically as:

$$\begin{aligned} W[A] = {} & P(\text{different value of } A \mid \text{nearest instances from different classes}) \\ & - P(\text{different value of } A \mid \text{nearest instances from the same class}) \end{aligned} \tag{2.8}$$

2.4.2 Analysis of Variance (ANOVA) based feature selection

ANOVA is a statistical method used to test the differences between two or more means [191, 192]. In the context of machine learning and feature selection, ANOVA is used to determine the relevance of individual features by analyzing the variance within and between groups of data corresponding to different classes. The key idea is that a feature is considered relevant if the means of different classes for that feature

are significantly different. ANOVA computes the variance ratio between the classes to the variance within the classes (the F-statistic). A higher F-statistic indicates that the feature has a significant impact on the target variable and is thus important for classification. The advantages of ANOVA include its simplicity, effectiveness in identifying significant features, and its ability to handle multiple classes simultaneously. ANOVA is particularly useful when the goal is to identify features that contribute to the differences between classes, thereby enhancing the performance of classification models. Algorithm 2 outlines the application of ANOVA for feature selection.

2.4.2.1 Equations and Variables

1. Between-Group Sum of Squares (BSS):

$$\text{BSS}_i = \sum_{k=1}^{K} n_k (\bar{X}_{ik} - \bar{X}_i)^2 \tag{2.9}$$

2. Within-Group Sum of Squares (WSS):

$$\text{WSS}_i = \sum_{k=1}^{K} \sum_{j=1}^{n_k} (X_{ijk} - \bar{X}_{ik})^2 \tag{2.10}$$

3. Between-Group Mean Square (BMS):

$$\text{BMS}_i = \frac{\text{BSS}_i}{K-1} \tag{2.11}$$

4. Within-Group Mean Square (WMS):

$$\text{WMS}_i = \frac{\text{WSS}_i}{N-K} \tag{2.12}$$

5. F-statistic:

$$F_i = \frac{\text{BMS}_i}{\text{WMS}_i} \tag{2.13}$$

2.4.2.2 Variable Explanations

- A_i: The i-th feature in the dataset.
- X_{ij}: The value of the i-th feature for the j-th instance.
- $\bar{X}_{ik}$: The mean value of the i-th feature for the k-th class.
- $\bar{X}_i$: The overall mean value of the i-th feature across all classes.
- K: The number of classes.
- n_k: The number of instances in the k-th class.
- N: The total number of instances in the dataset.
- BSS_i: The Between-Group Sum of Squares for the i-th feature.
- WSS_i: The Within-Group Sum of Squares for the i-th feature.
- BMS_i: The Between-Group Mean Square for the i-th feature.
- WMS_i: The Within-Group Mean Square for the i-th feature.
- F_i: The F-statistic for the i-th feature.

2.4.2.3 Interpretation

An F-statistic significantly larger than 1 indicates that the variance between the class means is greater than the variance within the classes, suggesting that the feature is relevant for distinguishing between classes. A high F-statistic value leads to rejecting the null hypothesis, implying that the feature has a significant impact on the classification.

Algorithm 2 ANOVA Algorithm for Feature Selection

Input: for each feature A_i in the dataset, values X_{ij} for each instance j and their class labels

Output: the F-statistic for each feature

1: **for** each feature A_i **do**

2: Calculate the mean of feature A_i for each class C_k: $\bar{X}_{ik}$

3: Calculate the overall mean of feature A_i: $\bar{X}_i$

4: Compute the Between-Group Sum of Squares (BSS):

$$\text{BSS}_i = \sum_{k=1}^{K} n_k (\bar{X}_{ik} - \bar{X}_i)^2 \tag{2.14}$$

5: Compute the Within-Group Sum of Squares (WSS):

$$\text{WSS}_i = \sum_{k=1}^{K} \sum_{j=1}^{n_k} (X_{ijk} - \bar{X}_{ik})^2 \tag{2.15}$$

6: Calculate the Between-Group Mean Square (BMS):

$$\text{BMS}_i = \frac{\text{BSS}_i}{K-1} \tag{2.16}$$

7: Calculate the Within-Group Mean Square (WMS):

$$\text{WMS}_i = \frac{\text{WSS}_i}{N-K} \tag{2.17}$$

8: Compute the F-statistic for feature A_i:

$$F_i = \frac{\text{BMS}_i}{\text{WMS}_i} \tag{2.18}$$

9: **end for**

2.4.3 Chi-square for Feature Selection

The Chi-square test is a statistical method used to determine if there is a significant association between categorical features and the target variable [193]. In the context of feature selection, the Chi-square test evaluates the independence of features and their relevance to the target class. A high Chi-square statistic indicates a strong association between the feature and the class, suggesting that the feature is relevant. Algorithm 3 outlines the application of Chi-square for feature selection.

Algorithm 3 Chi-square Algorithm for Feature Selection

Input: a dataset with categorical features A_i and class labels

Output: Chi-square statistic χ_i^2 for each feature

1: **for** each feature A_i **do**

2: **for** each class C_j **do**

3: Calculate the observed frequency O_{ij} of feature A_i in class C_j

4: Calculate the expected frequency E_{ij} of feature A_i in class C_j assuming independence:

$$E_{ij} = \frac{(\text{total instances in } A_i) \times (\text{total instances in } C_j)}{\text{total instances}} \tag{2.19}$$

5: **end for**

6: Compute the Chi-square statistic for feature A_i:

$$\chi_i^2 = \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{2.20}$$

7: **end for**

2.4.3.1 Equations and Variables

1. Expected Frequency:

$$E_{ij} = \frac{(\text{total instances in } A_i) \times (\text{total instances in } C_j)}{\text{total instances}} \tag{2.21}$$

2. Chi-square Statistic:

$$\chi_i^2 = \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{2.22}$$

2.4.3.2 Variable Explanations

- A_i: The i-th feature in the dataset.
- O_{ij}: The observed frequency of feature A_i in class C_j.
- E_{ij}: The expected frequency of feature A_i in class C_j.
- C_j: The j-th class.
- χ_i^2: The Chi-square statistic for the i-th feature.

2.4.3.3 Interpretation

A high Chi-square statistic suggests that the observed frequency of a feature in different classes deviates significantly from the expected frequency under the assumption of independence. This implies that the feature has a significant association with the class labels and is relevant for classification.

2.4.4 Mutual Information for Feature Selection

Mutual Information (MI) is a measure of the mutual dependence between two variables [194, 195]. In the context of feature selection, MI quantifies the amount of information obtained about the target variable through the feature, thus identifying relevant features for classification tasks. MI is particularly useful for capturing non-linear relationships between features and the target variable. Algorithm 4 outlines the application of Mutual Information for feature selection.

Algorithm 4 Mutual Information Algorithm for Feature Selection

Input: a dataset with features A_i and class labels C

Output: Mutual Information $MI(A_i; C)$ for each feature

1: **for** each feature A_i **do**

2: Calculate the joint probability distribution $P(A_i, C)$

3: Calculate the marginal probabilities $P(A_i)$ and $P(C)$

4: Compute the Mutual Information for feature A_i:

$$MI(A_i; C) = \sum_{a \in A_i} \sum_{c \in C} P(a, c) \log \frac{P(a, c)}{P(a)P(c)} \tag{2.23}$$

5: **end for**

2.4.4.1 Equations and Variables

1. Joint Probability Distribution:

$$P(a, c) = \frac{\text{Number of instances with } A_i = a \text{ and } C = c}{\text{Total number of instances}} \tag{2.24}$$

2. Marginal Probability:

$$P(a) = \frac{\text{Number of instances with } A_i = a}{\text{Total number of instances}} \tag{2.25}$$

$$P(c) = \frac{\text{Number of instances with } C = c}{\text{Total number of instances}} \tag{2.26}$$

3. Mutual Information:

$$MI(A_i; C) = \sum_{a \in A_i} \sum_{c \in C} P(a, c) \log \frac{P(a, c)}{P(a)P(c)} \tag{2.27}$$

2.4.4.2 Variable Explanations

- A_i: The i-th feature in the dataset.
- $P(A_i)$: The marginal probability of feature A_i.
- $P(C)$: The marginal probability of class C.
- $P(a, c)$: The joint probability of feature value a and class c.
- $MI(A_i; C)$: The Mutual Information between feature A_i and class C.

2.4.4.3 Interpretation

A high Mutual Information value indicates a strong dependency between the feature and the target variable, implying that the feature is highly relevant for classification. MI is non-negative and symmetric, meaning $MI(A_i; C) = MI(C; A_i)$.

2.5 Tools and Frameworks

This section details the primary tools and frameworks used in our research for recognizing static and dynamic hand gestures via surface electromyography (sEMG), as covered in Chapters 3 to 8. Our work mainly utilizes the open-source platforms Tensorflow and Keras for implementation.

2.5.1 Keras

Keras, released as open-source software in March 2015, is designed for building deep learning models [196]. It offers a Python-based interface for artificial neural networks and integrates with Tensorflow. Created by Google engineer François Chollet, the developer of the XCeption deep neural network model, Keras is tailored for rapid experimentation and is known for its user-friendliness, extensibility, and modularity. Beyond the standard deep learning functionalities, Keras also enables the deployment of deep learning models on both smartphones and web platforms.

2.5.2 Tensorflow

Developed by the Google Brain team in November 2015, Tensorflow is a comprehensive, free, and open-source software library for machine learning applications [197]. This library offers a broad range of functions suitable for various machine and deep learning tasks. It specializes in providing capabilities for distributing deep learning models and operates as a symbolic math library that supports data flow and differentiable programming. Tensorflow is utilized globally by numerous companies and research groups, including those working on projects like DeepDream, to develop and implement deep learning models.

2.5.3 Raspberry Pi 3

The Raspberry Pi 3 Model B is an economical and versatile single-board computer optimized for IoT applications [198]. It features a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, offering sufficient processing power for a range of IoT tasks. With 1 GB of LPDDR2 RAM, it supports multitasking and lightweight OS environments such as Raspbian and Ubuntu Core.

The board integrates 802.11n Wi-Fi and Bluetooth 4.1 for wireless communication, essential for IoT deployments. Its 40-pin GPIO header provides extensive I/O options, enabling direct interfacing with sensors, actuators, and other peripherals. Storage is handled via a microSD card slot, which serves as the boot medium, allowing for flexible and expandable storage solutions.

The Raspberry Pi 3 operates on a 5V micro-USB power supply, with low power consumption, making it suitable for energy-constrained IoT applications. Additionally, it offers multiple connectivity interfaces, including four USB 2.0 ports, HDMI, and a 3.5mm audio jack, facilitating various peripheral connections.