HAND GESTURE CLASSIFICATION USING SEMG SIGNALS

Dissertation Phase-2 Report

submitted by

Mr. ABHIN SHANKAR KURUP C. M.

REG NO : TKM22MEAI06

to

the APJ Abdul Kalam Technological University in partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY

in

Artificial Intelligence

CENTRE FOR ARTIFICIAL INTELLIGENCE

TKM COLLEGE OF ENGINEERING, KOLLAM

JUNE 2024

Thangal Kunju Musaliar College of Engineering
Centre for Artificial Intelligence

C E R T I F I C A T E

This is to certify that, this report titled ***HAND GESTURE CLASSIFICATION USING SEMG SIGNALS*** is a bonafide record of the **Dissertation Phase-2** presented by **ABHIN SHANKAR KURUP C. M. (TKM22MEAI06)**, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, **M. Tech in Artificial Intelligence** in **APJ Abdul Kalam University**.

Internal Supervisor	Project Coordinator	Head of the Department
Dr. Nissan Kunju Associate Professor Dept. of ECE TKMCE	Dr. Sumod Sundar Associate Professor Centre for AI TKMCE	Dr. Imthias Ahamed T. P. Professor Centre for AI TKMCE

ACKNOWLEDGEMENT

A successful project is a fruitful culmination of efforts by many people, some directly involved and some others indirectly, by providing support and encouragement. Firstly I would like to thank the almighty for giving me the wisdom and grace for making my project a memorable one. I thank him for steering me to the shore of fulfillment under his protective wings.

I express my sincere gratitude to **Dr. T. A. Shahul Hameed**, Principal of T.K.M College of Engineering for allowing me to present my project. I would like to thank **Dr. Imthias Ahamed T. P.**, Professor and Head of the Department, Centre for Artificial Intelligence, TKM College of Engineering, Kollam, for his constant support and encouragement throughout the work.

I express my gratitude to the Department of Electronics and Communication Engineering for the opportunity to work in the Biomedical Engineering Research Lab.

With a profound sense of gratitude, I would like to express my heartfelt thanks to my internal supervisor, **Dr. Nissan Kunju**, Associate Professor, Department of ECE, **Mr. Kartik S. Prakash** Research scholar ,Biomedical Engineering Research Lab, Department of ECE and Project Coordinator, **Dr.Sumod Sundar**, Associate Professor, Center for AI, TKM College of Engineering for their expert guidance, cooperation, and immense encouragement. I also extend my thanks to the entire faculty and staff members of the Centre for AI and Department of ECE, TKMCE, who have encouraged me throughout this work.

I also express my thanks to my loving parents and friends, for their support and encouragement in the successful completion of this work.

ABHIN SHANKAR KURUP C. M.

Abstract

This work presents the development of a robust hand gesture classification system based on surface Electromyography (SEMG) signals, designed for real-world applications. The project utilized data collected from subjects wearing electrodes on their hands, with initial investigations conducted using the wrist electrode data from GrabMayo dataset.Later on used statistical and anatomical knowledge to find the best electrode placement positions so that amount of data needed and complexity of model are reduced. Preprocessing techniques were employed to convert the raw SEMG signals into mel spectrogram images, alongside extraction of time domain features. These features were then utilized to train and fine-tune machine learning models including Support Vector Machines (SVM), Random Forest, and K-Nearest Neighbors (KNN). Subsequently, the study explored deep learning approaches, utilizing both pretrained neural networks and a custom Convolutional Neural Network (CNN) architecture. The investigation aimed to increase classification accuracy while reducing model complexity to facilitate implementation on wearable devices. Comparative analysis of the machine learning and deep learning models was performed using performance metrics. Additionally, a robust hand gesture classifier model with small size was developed with good accuracy and low latency in prediction. The findings of this investigation can contribute to the advancement of gesture recognition systems, offering insights into effective deep learning approaches for SEMG-based hand gesture classification in real-world scenarios.

Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

AR	Augmented Reality
VR	Virtual Reality
SVM	Support Vector Machine
CNN	Convolutional Neural Network Nm
KNN	K-Nearest Neighbors
SEMG	surface Electromyography
HGR	Hand Gesture Recognition
HCI	Human-Computer Interaction
RGB	Red Green Blue
PNG	Portable Network Graphics
SSC	Slope Sign Change
RMS	Root Mean Square
MAV	Mean Absolute Value
MAR	Mean Average Recall
MAP	Mean Average Precision
DCT	Discrete Fourier Transform
MFCC	Mel Frequency Cepstral Coefficients
VGG	Visual Geometry Group
CBAM	Convolutional Block Attention Module
RELU	Rectified Linear Unit

Chapter 1

Introduction

Surface Electromyography (SEMG) stands at the forefront of technological innovation in the realm of biomedical engineering, providing a unique window into the electrical dynamics of muscles. This non-invasive technique involves strategically placing electrodes on the skin to capture and record the electrical signals emanating from muscle fibers during contractions. The versatility of SEMG has propelled it into various domains, with particular emphasis on its applications in Human-Computer Interaction (HCI). Gesture recognition through SEMG has garnered substantial attention, offering a direct reflection of the functional state of human nerves and muscles. As the signals are inherently non-stationary, representing the amalgamation of subcutaneous athletic action potentials, their potential extends to advanced prosthetics control, Hand Gesture Recognition (HGR), and biometric security systems..

This project holds significant importance as it addresses critical challenges and opportunities at the intersection of biomedical engineering, prosthetics, and Human-Computer Interaction (HCI). By focusing on the utilization of Surface Electromyography (SEMG) signals for advanced prosthetics control and gesture recognition, the project aims to enhance the quality of life for individuals with limb loss, offering them improved control over artificial limbs. The exploration of time and frequency domain features of SEMG signals not only contributes to the refinement of assessments in sports medicine, physical therapy, and bio-mechanics but also promises to advance the broader field of biomedical applications. Moreover, the project's emphasis on feature extraction and recognition aims to bridge the gap between research and practical implementation, fostering the development of more reliable and user-friendly systems. In doing so, this project not only addresses immediate challenges in healthcare and technology but also contributes valuable insights to the scientific community, potentially inspiring future innovations and shaping the trajectory of research in the field.

A key evaluation measures like accuracy, latency , generalizability , were used to evaluate the model's performance. This project is important not only for its immediate applications in prosthetics, HCI, IoT, and biomedical engineering but also for its potential to drive advancements in related fields, improve the quality of patient care, and inspire future research and innovation.

This project is expected to produce results that are not limited to academics. Practical

applications of the study includes far-reaching implications in prosthetics advancement, intuitive Human-Computer Interaction, and personalized rehabilitation programs. By decoding the intricate relationship between Surface Electromyography (SEMG) signals and muscle movements, the research contributes to fields such as sports medicine, bio-mechanics, and assistive technologies, promising improved healthcare, enhanced athletic performance, and innovative solutions for individuals with mobility challenges.

1.1 Objectives

- To reduce number of electrodes required for making a wearable device like wrist watch and use only the data from electrodes placed near wrist to classify hand gestures.
- Develop a custom model for classification of spectral images .
- Train the data on machine learning models (after feature extraction) as well as deep learning models (spectral image).
- Evaluate the performance of the proposed model and other models trained for signal classification and do a comparative analysis of various models.

1.2 Research Gap

Several studies were conducted on self collected or prominent datasets like Ninapro or Grab-Mayo but the following gaps were found out during literature survey prompted this work.

- Most models developed for classification were huge with billions of parameters,this huge size made them slow and less useful in real life scenarios.
- Researchers were producing high accuracy classification models but they used data from a large number of electrodes placed at various part of forearm so the accuracy came at the expense of cost and complexity.

Chapter 2

Literature Survey

Hands are necessary for carrying out daily tasks like reaching for a drink or communicating with others by waving farewell. As the world's population ages, neurological illnesses are becoming more common, which is resulting in a loss of hand function and a decline in quality of life [1].Automated hand gesture recognition can be coupled with an orthosis [2] to enhance grasp strength, or it can be incorporated with games [3] to evaluate rehabilitation progress through active involvement.Gesture recognition interfaces can decode human intention orders for prosthetic manipulation movement control [4] or grasping force control [5], enabling independent living. Similarly, upper extremity amputees frequently maintain intention and neural motor control which can be amplified. [6].

Qureshi et.al worked on classification of hand gestures using spectral images [7] and were able to design a custom network.They introduced a unique approach using Mel spectrogram images for SEMG signal classification and compared the proposed CNN with existing methods.They achieved an accuracy of 99.42% $\pm$ 0.42% for the able-bodied (healthy and nondisabled) subjects and 98.00% $\pm$ 0.58% for the amputee subjects in the within-day analysis. These results demonstrate the high classification accuracy of the CNN model for both short-term and long-term surface electromyography (SEMG) classification of hand motions. Additionally, the CNN outperformed previous Stacked Sparse Autoencoders (SSAE) and pretrained transfer learning (TL) models, showcasing its effectiveness in SEMG classification.

In addition to facilitating daily activities at home, these hand gesture recognition interfaces lighten the workload for hospital-based specialty specialists. For the hearing impaired, hands are the major means of communication [8] and automatic sign language translation can be used to facilitate communication with the unimpaired through hand gesture recognition interfaces. Additionally, hand gesture recognition has demonstrated promise for more natural communication for a range of new applications involving human-computer interaction [2], such as gesture interaction with smartphones [4], In VR games, hand gestures can be used to perform actions such as picking up objects, throwing, pointing, or casting spells. This makes the gaming experience more immersive and engaging.gesture recognition can offer alternative ways to control and navigate AR systems [9], and in-car menu control to prevent the need to look for controls while operating a vehicle [3].

Better materials, innovative sensing methods, and embedded system downsizing can make

wearable interfaces more user-friendly and intuitive, while machine learning algorithm advancements could lead to more robust, powerful, and accurate tracking and classification capabilities.The ability of algorithms to recognize hand gestures has advanced dramatically in the last several years. Prior methods that relied mostly on human expertise, such as fuzzy logic or simple threshold control, have been replaced in recent years by machine learning, which includes statistical learning methods such maximal a posteriori and expectation maximization [3].

In order to improve performance and solve biological difference problems without relying on apriori knowledge, deep learning techniques [6], which are widely used for image classification, are also emerging in hand gesture recognition applications. These techniques include convolutional neural networks (CNN) , transfer learning [5], and meta learning [8].Meta-learning for gesture classification in VR involves training a model that can quickly adapt to new gestures or environments with minimal data.

Chapter 3

Methodology

Two approaches were used in this work first one machine learning based approach which requires relevant features to be extracted and saved to a features.mat file. Second approach(deep learning based) follows same procedure but instead of feature extraction mel-spectrum was calculated and stored as a 227*227 RGB image in PNG format. Once studies on 6 channels(electrodes at wrist) were complete we moved on to reducing the number of electrodes requires while keeping a balance between model complexity and required accuracy. Anatomical studies and statistical observations were used to support the selection of best electrodes.

3.1 Data Set

The dataset used for study was GrabMayo dataset [10] .It contains signal data of 16 gestures with rest as the 17th gesture. The gestures in consideration can be seen in Figure 3.1 [10]

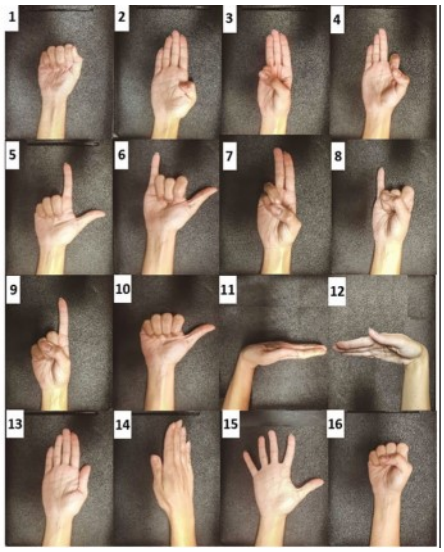


Figure 3.1: Gestures For Classification

The dataset was acquired as three sessions spanning three days to accommodate maximum diversity. During each session 43 persons were asked to perform 17 gestures and repeat the

trial for each gesture 7 times. Sampling frequency was 2048 Hz,which goes way beyond the minimum threshold set by nyquist's theorem. Dataset structure with dimensions can be seen in Figure 3.2. Thirty two mono polar electrodes(16 on wrist and 16 on forearm) placed at preset distance as shown in Figure 3.3[10] were used to acquire data.

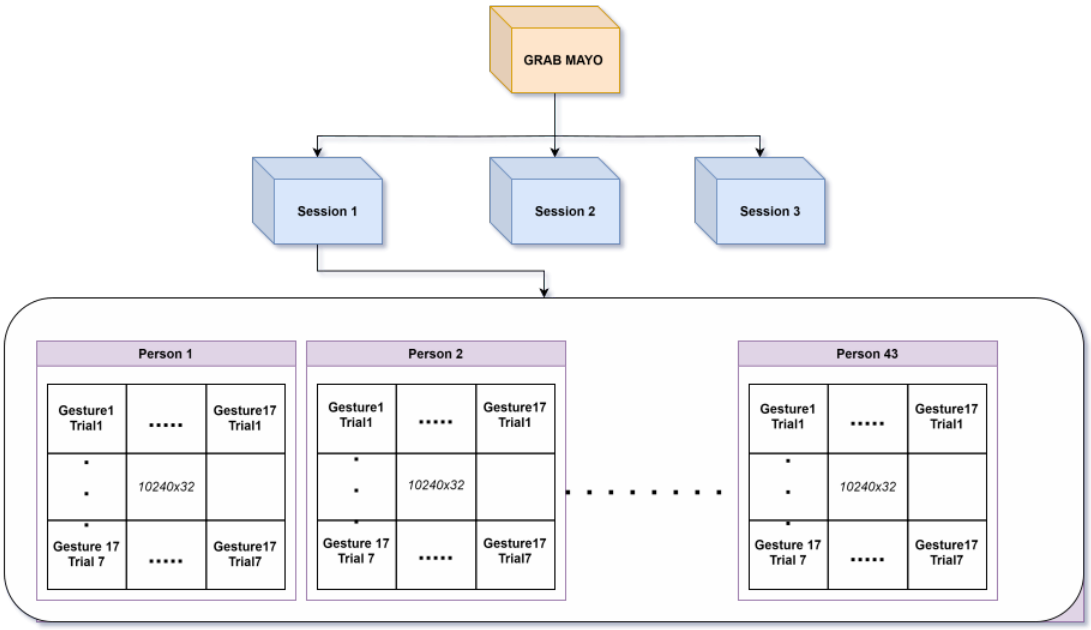


Figure 3.2: Dataset Structure

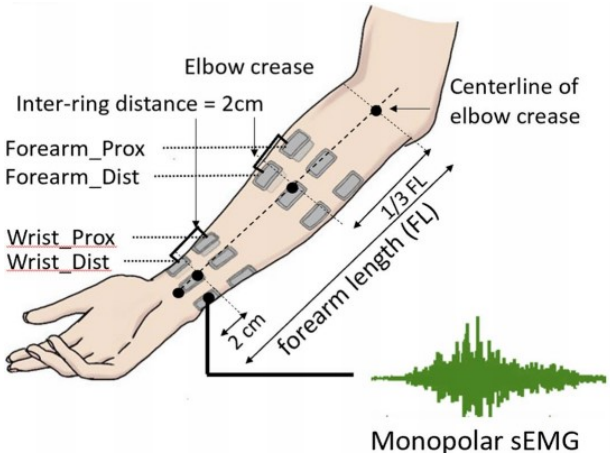


Figure 3.3: Location of Electrodes

3.2 Pre-processing

3.2.1 Creating the 'Complete Set' Matrix

Figure. 3.4 shows the method of combining data to form the complete set. The 'Complete Set' matrix is structured as a 43x17 matrix, where each cell represents a single gesture of

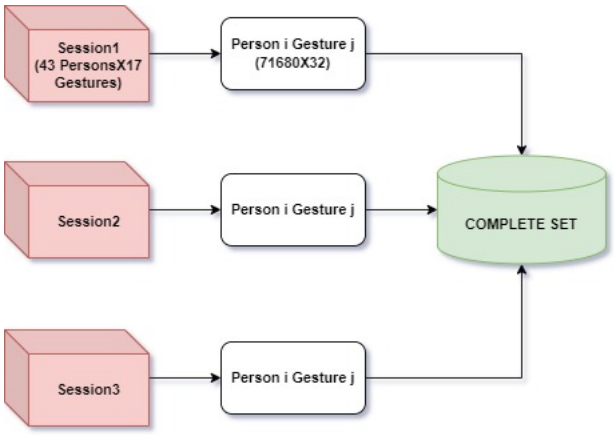


Figure 3.4: Pre-processing to Create Complete Set

a person. The dimensions of each cell within 'Complete Set' are 71680x32, where 71680 is the data aggregated from 7 trials of each gesture. Each column in this matrix represents the reading of one electrode for a gesture.

3.2.2 Channel Configuration

Data set consists of 32 channels among them only wrist electrodes will be used.

- Forearm Channels: F1-F8 and F9-F16
- Wrist Channels: W1-W6 and W7-W12
- Unused Channels: U1-U4 This project is based on wrist channels.

3.2.3 Data Conversion to Bipolar

Mono-polar data undergoes conversion to bipolar format by subtracting the value of the m+8th electrode from that of the mth electrode. This transformation yields a new Data matrix with expanded dimensions of 71680x6.[10]

3.2.4 Segmentation

Data segmentation is a process where the dataset is partitioned into segments, each comprising 512 data points, corresponding to a 250ms window. These segments are created with an overlap of 256 samples, meaning that the last 256 samples of each window are also included in the following segment. This approach not only ensures continuous data coverage but also effectively mitigates the risk of data loss.

3.2.5 Feature Extraction

Creation of the complete matrix is followed by windowing stacking and feature extraction.Since the data is of mono polar channels we also convert them to bipolar data in between. The summary is shown in Figure 3.5 Once dataset is preprocessed now it is ready for feature

extraction. Features are extracted for each window (512 samples) for every channel. The following features are calculated and appended to the feature metrics,

- Mean Absolute Value
- Slope Sign Change
- Root Mean Square Value
- Zero Crossing

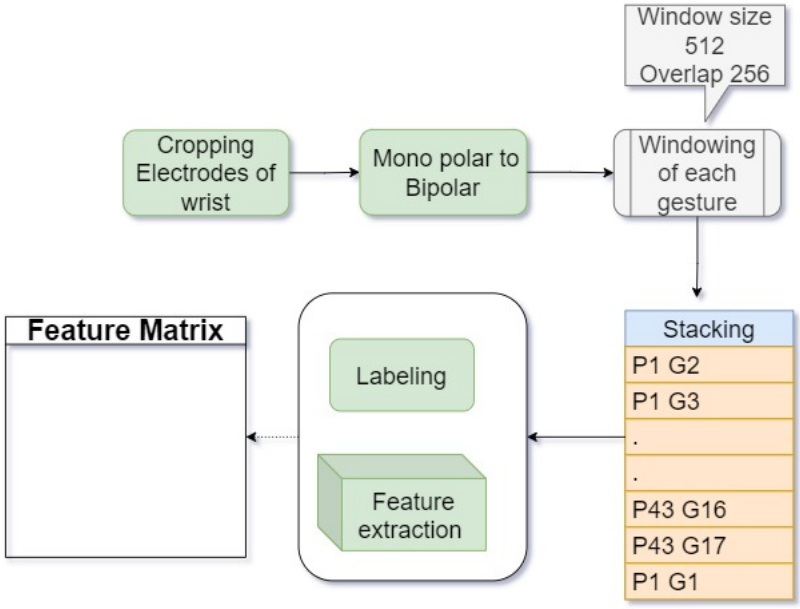


Figure 3.5: Pre-processing pipeline after creation complete matrix

The feature metrics have a shape of 24x(number of windows of a gesture), capturing features for each channel and each window.

Slope Sign Change

SSC is valuable in gesture recognition tasks where distinct patterns of muscle activation are associated with specific movements. Its ability to detect changes in signal direction enhances the feature set for accurate classification of diverse gestures.

$$\text{SSC} = \sum_{i=2}^{N-1} \begin{cases} 1 & \text{if } (\text{sgn}(x_i - x_{i-1}) \neq \text{sgn}(x_{i-1} - x_{i-2})) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Mean Absolute Value

The relevance of MAV in EMG signal processing lies in its ability to capture the overall magnitude of the muscle activation. Higher MAV values indicate a greater average muscle

activity, while lower MAV values suggest lower muscle activity. MAV is often used as a feature in EMG pattern recognition and classification tasks, where different muscle activities are associated with distinct MAV patterns.

$$MAV = \frac{1}{n}\sum_{i=1}^{n}|x_i| \tag{3.2}$$

RMS Value

Root Mean Square (RMS) in EMG signals provides a measure of overall signal amplitude, capturing both positive and negative excursions. It is crucial for assessing muscle activity levels, detecting variations across trials, and ensuring signal stability. RMS serves as a key feature for signal classification tasks, aiding in the development of models for applications like gesture recognition and muscle fatigue assessment. Its versatility makes it valuable in diverse fields, including rehabilitation, prosthetics, and human-computer interaction.

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2} \tag{3.3}$$

Zero Crossing

It refers to number of times emg signal changing from negative to positive and vice versa.

$$ZC = \frac{1}{2}\sum_{i=2}^{N}\left|\frac{1}{2}(\text{sgn}(x_i) - \text{sgn}(x_{i-1}))\right| \tag{3.4}$$

3.2.6 Labeling

Labels are assigned to each window based on the rightmost column, where '1' represents the 1st gesture, '2' represents the 2nd gesture, and so on. Overall framework for machine learning part is shown in the 3.6

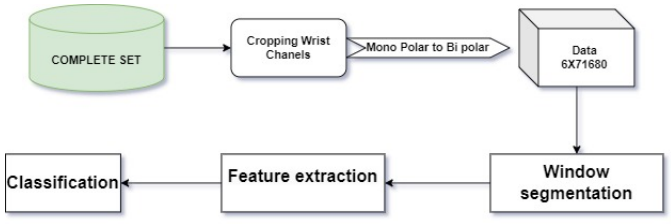


Figure 3.6: Pre-processing pipeline for Machine learning

3.3 Machine Learning Based Technique

Figure 3.7 Shows the methodology adopted for machine learning based technique. The research begins its investigation from dataset(GrabMyo),consists of 43 participants (sub-

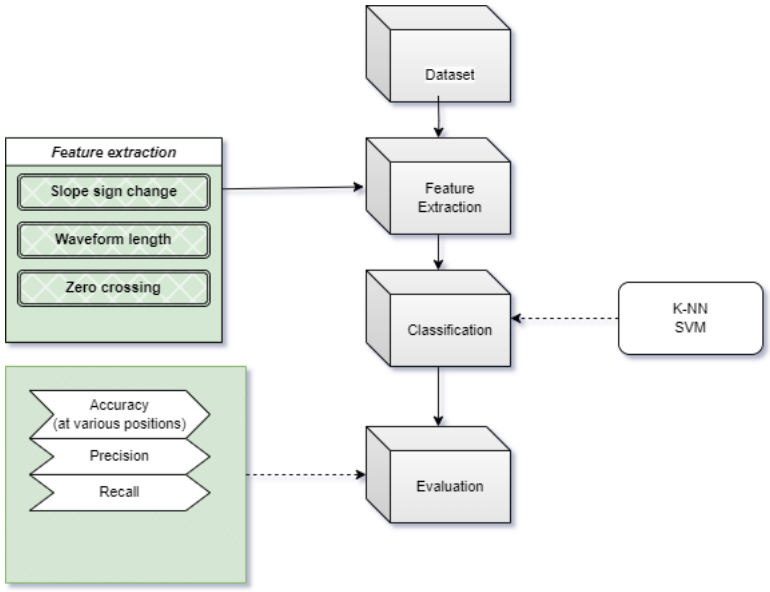


Figure 3.7: Machine Learning Methodology

sequently termed as users), three sessions in three separate days (subsequently termed as sessions) of data collection, 16 hand and finger gestures each with seven repetitions (subsequently termed as trials). This dataset is the largest SEMG dataset in terms of the total number of recording sessions (43 users×3 days=129 recording sessions) are the basic raw data that forms the basis of the work of categorization that follows. The dataset gathered were then go through preprocessing steps which are implemented to enhance the quality of the collected SEMG signals. The preprocessing includes band pass filter,window segmentation and notch filter.The application of a band pass filter is crucial to eliminate noise and focus on the relevant frequency range associated with muscle activity. Window segmentation is employed to break down the continuous signal into manageable segments, facilitating the analysis of specific gestures. Additionally, a notch filter is utilized to further eliminate interference from specific frequencies.

Following preprocessing, the feature extraction stage plays a pivotal role in capturing distinctive characteristics of the SEMG signals that are indicative of different hand gestures. Features such as slope sign change, waveform length, and zero crossing are extracted to quantify aspects of the signal dynamics, amplitude variations, and frequency content, respectively. These features collectively provide a comprehensive representation of the muscle activation patterns during various hand movements.Once the feature extraction is complete, the model proceeds to the classification phase. Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) classifiers are employed to learn and categorize the SEMG signals into different hand gesture classes. These classifiers leverage the extracted features to distinguish between the nuanced patterns associated with each gesture. The model's performance is measured by the evaluation metrics, which include the confusion matrix, precision, recall, and F1 score. All of these measures provide a thorough grasp of the model's precision and ability to accurately categorise signals with respect to their corresponding gestures. The process culminates in the model being applied to test data that has not yet been seen, and it ends with a thorough evaluation of the model's classification accuracy. The final stan-

dard by which the model's performance is measured against the defined evaluation metrics is the unseen data. All things considered, this created process guarantees a methodical and comprehensive approach.

3.3.1 Feature extraction

Features like RMS value,Mean Absolute value,Zero Crossing,Slope sign change were calculated for each electrode after windowing(sample size 512).For each each electrodes 512 values these 4 features were extracted. The following models are used for the experimentation.

3.3.2 Training

The extracted features were passed for training to machine learning algorithms namely KNN(which is known for its light weight) and SVM(known for its better performance)

3.3.3 Support Vector Machine(SVM)

.

In this study, a support vector machine (SVM) classifier is defined for hand gesture classification task. The SEMG signals, collected from strategically placed electrodes, undergo a preprocessing phase involving band pass filtering, window segmentation, and notch filtering to enhance signal quality. Following this, relevant features, such as slope sign change, waveform length, and zero crossing, are extracted from the preprocessed signals. These features serve as the input data for the SVM classifier. Unlike neural networks, SVM excels in high-dimensional spaces, making it well-suited for the intricate nature of SEMG data. The SVM model is configured to find an optimal hyperplane that effectively separates different hand gestures in the feature space. Parameters like the choice of kernel and regularization are carefully tuned. The SVM classifier is then trained on a labeled dataset, where each data point corresponds to a specific hand gesture, and its performance is rigorously evaluated using metrics like accuracy and precision. The system also takes into consideration individual variability in muscle anatomy and SEMG signal characteristics, ensuring adaptability across diverse users. Ultimately, the trained SVM model becomes a reliable tool for real-time hand gesture classification in human-computer interaction scenarios, offering precision and robustness in interpreting user intent through SEMG signals.

3.3.4 K-Nearest Neighbors(KNN)

In this study, a K-Nearest Neighbors (KNN) classifier is defined for hand gesture classification task.The journey begins with the collection of SEMG signals through strategically placed electrodes, followed by a meticulous preprocessing phase and feature extraction process.Unlike more complex models, KNN is embraced for its simplicity and effectiveness, particularly in scenarios where interpretability is paramount. The KNN classifier operates without an explicit training phase, relying on the memorization of the entire dataset. Each SEMG signal's classification is determined by the majority class among its k-nearest neighbors in the feature space. Parameters like the choice of 'k' (number of neighbors) are considered, influencing the granularity of the classification decision. The KNN model is trained on a labeled dataset, and its performance is rigorously evaluated, commonly using metrics like

accuracy and precision. This classifier's straightforwardness makes it an appealing choice, especially in cases where the relationship between SEMG signal patterns and hand gestures is complex but can be effectively captured through proximity-based classification. As the KNN model demonstrates its adaptability and ease of implementation, it offers a valuable solution for real-time hand gesture classification in human-computer interaction scenarios, catering to applications where simplicity and interpretability are pivotal.

3.3.5 Performance Metrics

It is common to use precision(the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions) recall(a metric that measures how often a machine learning model correctly identifies positive instances (true positives) from all the actual positive samples in the dataset.) F1 Score and ROC character to evaluate a binary classification problem.Since the problem at hand is a multi class classification problem with 17 classes we have to adapt a modified methodology.Precision and Recall were calculated for each class and then averaged. Similarly F1 score was also calculated.

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

The Mean Average Precision (MAP) can be calculated as:

$$\text{MAP} = \frac{1}{N} \sum_{k=1}^{N} \text{AP}_k \tag{3.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.7}$$

The Mean Average Recall (MAR) can be calculated as:

$$\text{MAR} = \frac{1}{N} \sum_{k=1}^{N} \text{AR}_k \tag{3.8}$$

The weighted average of precision and recall is the F1-score. It considers both false positives and false negatives to determine the model's overall accuracy.

$$F1score = 2 * \frac{\text{Precision } * \text{ Recall}}{\text{Precision } + \text{ Recall}} \tag{3.9}$$

Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model. It summarizes the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model. These metrics are often used to assess the model's accuracy, precision, recall, and F1-score, among others. Here's how the confusion matrix is structured:

- True Positives (TP): These are cases where the model correctly predicted the positive class.

- True Negatives (TN): These are cases where the model correctly predicted the negative class.
- False Positives (FP): These are cases where the model incorrectly predicted the positive class when it should have been negative (e.g., falsely diagnosing a disease when it's not present). Also known as Type I error.
- False Negatives (FN): These are cases where the model incorrectly predicted the negative class when it should have been positive (e.g., failing to diagnose a disease when it's actually present). Also known as Type II error.

A confusion matrix helps assess the model's performance in terms of correctly classifying instances and identifying any biases it may have toward certain classes. From the confusion matrix, various performance metrics like accuracy, precision, recall, F1-score, and the Matthews correlation coefficient can be calculated to provide a comprehensive evaluation of the model's effectiveness. .

3.4 Deep Learning Based Technique

After completing the machine learning stage on 6 bipolar wrist electrodes accuracy obtained were good enough so that it doesn't justify the use of deep learning for classification. Now it is time to address the next challenge by reducing the amount of data required for classification.As mentioned in the objectives this can be achieved by selecting the best electrodes that explains the class of gesture. Figure 3.8 shows the steps involved in this part of the work. The deep learning part was performed after the selection of best wrist electrodes (4 out of 6) Two studies (anatomical and statistical analysis) which are synergistic for this purpose but entirely different were adopted to achieve this without going through expensive trial and error based training loop.

From the selected electrodes mel spectral images were obtained for each 512 samples and it was used as dataset for transfer learning with networks like alex net vgg16 mobilenet etc. The same dataset of melspectral images were used for designing a faster CustomNetwork with small number of parameters (as compared to transfer learned models).

3.4.1 Melspectrogram

The Mel Frequency Cepstral Coefficients (MFCCs) are a feature widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980's and have been state-of-the-art ever since. The equation for computing MFCCs is given by:

$$MFCC_i = \sum_{k=1}^{N} \log \left(\sum_{n=1}^{N} |X_k(n)|^2 H(n) \cos \left[i \frac{\pi}{N} (n + \frac{1}{2}) \right] \right) \cos \left[i \frac{\pi}{N} (k + \frac{1}{2}) \right] \tag{3.10}$$

where N is the number of samples, $X_k(n)$ is the Discrete Fourier Transform (DFT) of the signal at frame k and frequency bin n, $H(n)$ is the Mel filterbank, and i represents the coefficient index.

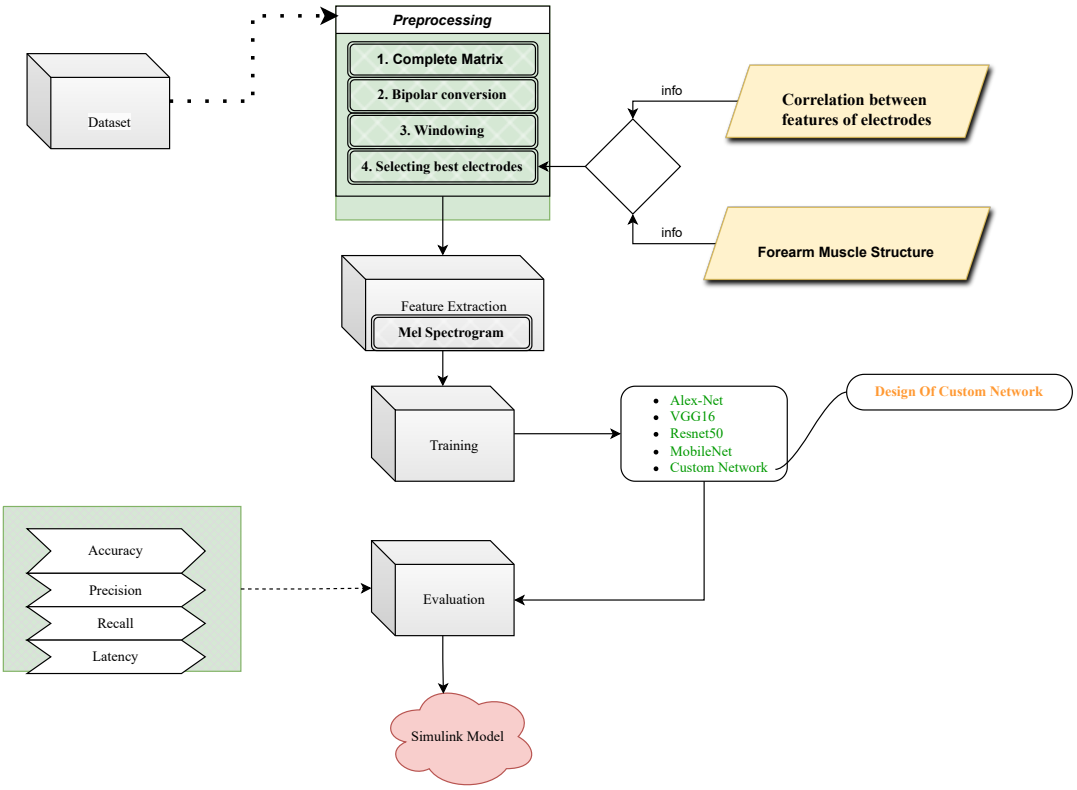


Figure 3.8: Deep Learning Methodology

MFCCs are derived from the Mel scale, which approximates the human ear's response to different frequencies. The Mel scale is a perceptual scale of pitches, which corresponds closely to the physical scale of frequencies. It's based on psycho-acoustic experiments that measure how humans perceive the distance between two tones.

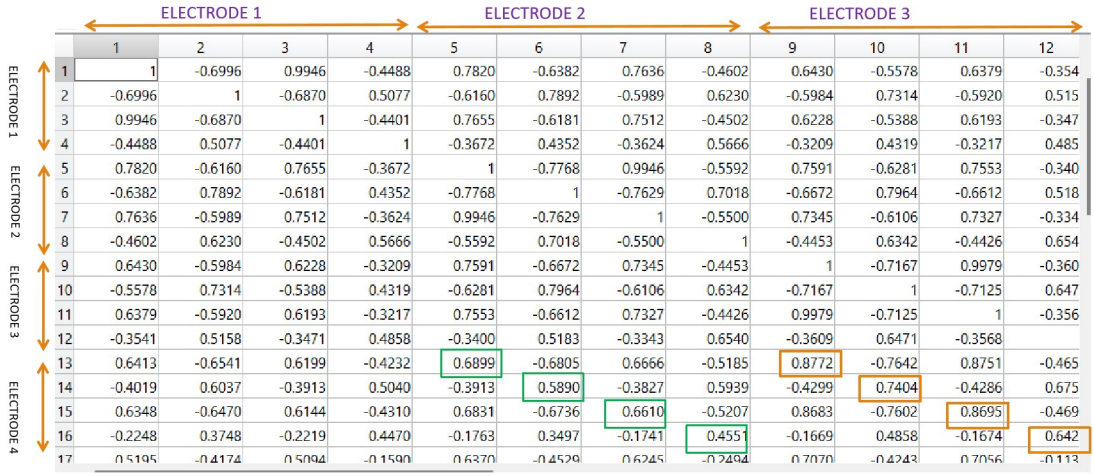
The MFCCs are obtained by taking the discrete cosine transform (DCT) of the log of the power spectrum of the signal. This process results in a set of coefficients that capture the spectral envelope of the signal.

MFCCs are used as features in many speech and audio processing tasks, such as speech recognition, speaker identification, and emotion recognition. They're particularly effective because they capture relevant information about the spectral characteristics of the signal while discarding irrelevant details. Additionally, they have been shown to be robust to noise and speaker variability, making them suitable for real-world applications.

3.4.2 Anatomical Study

To remove the redundant electrode we could use anatomical knowledge of forearm muscle groups which can be divided mainly to 2 sections flexor compartment and extensor compartment.From them using the knowledge of insertion and relevance following observations are made.

- Some muscles like palmaris longus were not present for every human being.
- Some muscles are important but they are deep inside.



	1	2	3	4	5	6	7	8	9	10	11	12
1	1	-0.6996	0.9946	-0.4488	0.7820	-0.6382	0.7636	-0.4602	0.6430	-0.5578	0.6379	-0.354
2	-0.6996	1	-0.6870	0.5077	-0.6160	0.7892	-0.5989	0.6230	-0.5984	0.7314	-0.5920	0.515
3	0.9946	-0.6870	1	-0.4401	0.7655	-0.6181	0.7512	-0.4502	0.6228	-0.5388	0.6193	-0.347
4	-0.4488	0.5077	-0.4401	1	-0.3672	0.4352	-0.3624	0.5666	-0.3209	0.4319	-0.3217	0.485
5	0.7820	-0.6160	0.7655	-0.3672	1	-0.7768	0.9946	-0.5592	0.7591	-0.6281	0.7553	-0.340
6	-0.6382	0.7892	-0.6181	0.4352	-0.7768	1	-0.7629	0.7018	-0.6672	0.7964	-0.6612	0.518
7	0.7636	-0.5989	0.7512	-0.3624	0.9946	-0.7629	1	-0.5500	0.7345	-0.6106	0.7327	-0.334
8	-0.4602	0.6230	-0.4502	0.5666	-0.5592	0.7018	-0.5500	1	-0.4453	0.6342	-0.4426	0.654
9	0.6430	-0.5984	0.6228	-0.3209	0.7591	-0.6672	0.7345	-0.4453	1	-0.7167	0.9979	-0.360
10	-0.5578	0.7314	-0.5388	0.4319	-0.6281	0.7964	-0.6106	0.6342	-0.7167	1	-0.7125	0.647
11	0.6379	-0.5920	0.6193	-0.3217	0.7553	-0.6612	0.7327	-0.4426	0.9979	-0.7125	1	-0.356
12	-0.3541	0.5158	-0.3471	0.4858	-0.3400	0.5183	-0.3343	0.6540	-0.3609	0.6471	-0.3568	
13	0.6413	-0.6541	0.6199	-0.4232	0.6899	-0.6805	0.6666	-0.5185	0.8772	-0.7642	0.8751	-0.465
14	-0.4019	0.6037	-0.3913	0.5040	-0.3913	0.5890	-0.3827	0.5939	-0.4299	0.7404	-0.4286	0.675
15	0.6348	-0.6470	0.6144	-0.4310	0.6831	-0.6736	0.6610	-0.5207	0.8683	-0.7602	0.8695	-0.469
16	-0.2248	0.3748	-0.2219	0.4470	-0.1763	0.3497	-0.1741	0.4551	-0.1669	0.4858	-0.1674	0.642
17	0.5195	-0.4174	0.5094	-0.1590	0.6370	-0.4529	0.6245	-0.2494	0.7070	-0.4243	0.7056	-0.113

Figure 3.9: Correlation matrix

- Eventhough Pronators and Suppinators are important,they play lesser role for making the gestures in consideration.
- Some muscle groups of hypothenar eminence were less important as the small finger dosen't play a major role in many gestures.
- Many muscles of thenar eminence don't go to forearm.

3.4.3 Statistical Study

Statistical study was based on the idea of removing highly interdependent electrodes which give out similar values for different features. 4 different features namely RMS, Mean Square, Slope Sign Change,Zero Crossing, were calculated for each of the 6 electrodes. and then correlation matrix was formed.As shown in Figure 3.9 The orange boxes highlights highly correlated electrodes 3 and 4 which implies we can avoid one of them from our further analysis without affecting result badly.

3.5 Deep Learning Architectures

3.5.1 Alex Net

AlexNet [11] is a convolutional neural network architecture that gained significant attention after winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It consists of eight layers, including five convolutional layers and three fully connected layers. Its architecture features novel aspects such as the extensive use of ReLU activation functions and dropout regularization to prevent overfitting. AlexNet's success marked a breakthrough in deep learning, demonstrating the power of convolutional neural networks for image classification tasks. Beyond image classification, AlexNet's architecture (Figure. 3.10[11]) has found applications in various computer vision tasks such as object detection, image segmentation, and even in non-visual domains like natural language processing. Its efficient design and remarkable performance have made it a foundational model in the field of deep learning, inspiring numerous subsequent architectures and research developments.

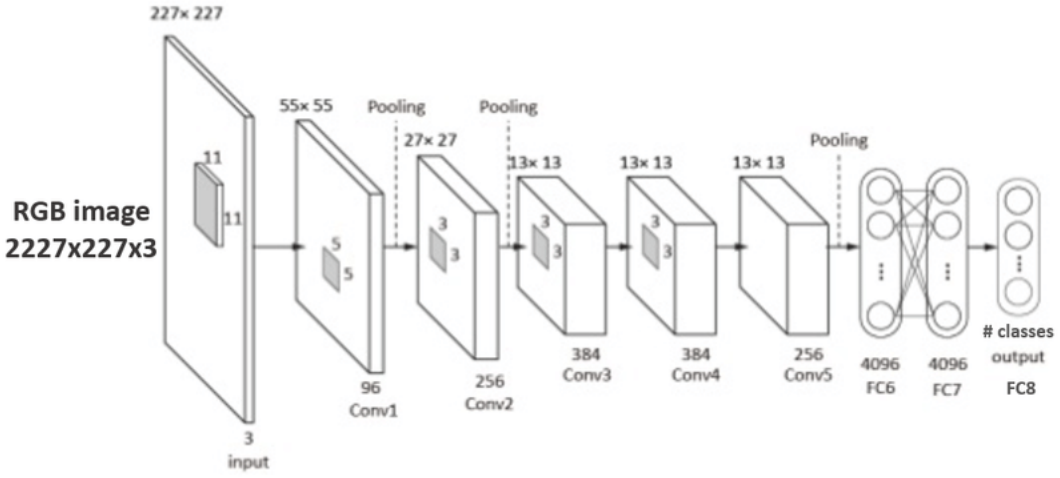


Figure 3.10: AlexNet Architecture

3.5.2 VGG 16

VGG 16 [12] is a convolutional neural network architecture that was proposed by the Visual Geometry Group (VGG) at the University of Oxford in 2014. It is characterized by its depth, consisting of 16 layers, including 16 convolutional layers and 3 fully connected layers. VGG 16's architecture (Figure. 3.11 [12]) maintains a simple and uniform structure, with small 3x3 convolutional filters and max-pooling layers used throughout. Despite its depth, VGG 16 remains relatively easy to understand and implement compared to more complex architectures. It achieved state-of-the-art results on the ImageNet dataset at the time of

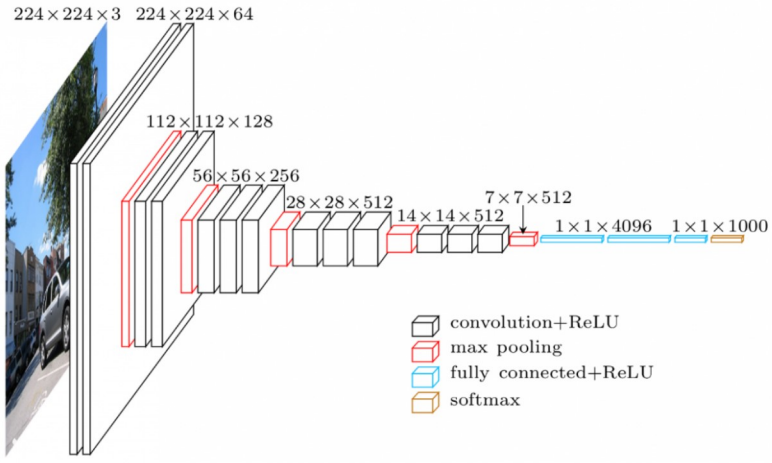


Figure 3.11: VGG 16 Architecture

its introduction, demonstrating its effectiveness in image classification tasks. Beyond image classification, VGG 16 has been widely adopted as a feature extractor in various computer vision applications, including object detection, image segmentation, and style transfer. Its

architecture's versatility and strong performance have made it a popular choice in both academic research and practical applications in the field of deep learning.

3.5.3 Resnet 50

ResNet, short for Residual Network, is a groundbreaking convolutional neural network architecture proposed by Microsoft Research in 2015 [13]. It addresses the problem of vanishing gradients in deep neural networks by introducing skip connections or residual connections. These connections allow information to bypass one or more layers, facilitating the training of very deep networks. ResNet architectures come in several variants, with ResNet-50, ResNet-101, and ResNet-152 being popular choices. These variants differ in the number of layers, with ResNet-50 having 50 layers and ResNet-152 having 152 layers. ResNet achieved state-of-the-art performance on various computer vision tasks, including image classification, object detection, and image segmentation, and has been widely adopted in both research and industry. Its success has inspired the development of other architectures based on skip connections, contributing to the advancement of deep learning.Figure. 3.12 [13]

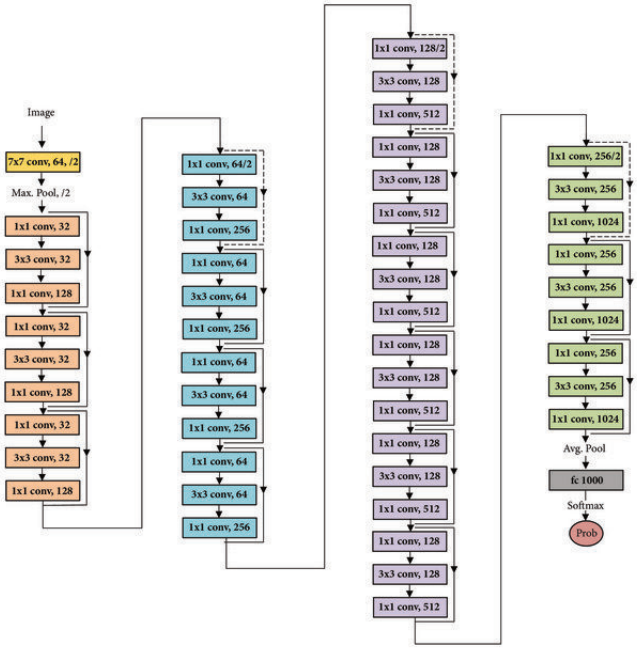


Figure 3.12: Resnet 50 Architecture

3.5.4 Xception

Xception [14] is a convolutional neural network architecture introduced by Google researchers in 2017. It stands for "Extreme Networks inception," and it is inspired by the Inception architecture but with a key difference: Xception replaces standard convolutional layers with depthwise separable convolutions. This modification significantly reduces the number of parameters and computations required while maintaining expressive power. Xception achieves state-of-the-art performance on various computer vision tasks, including image classification,

object detection, and image segmentation. Its architecture promotes better feature learning and efficiency compared to previous models, making it particularly suitable for resource-constrained environments. Xception has become a popular choice in both academic research and industry applications, contributing to the advancement of deep learning techniques for visual recognition tasks.

3.5.5 Custom Network

To make this Custom Network 3 separate blocks along with conventional CNN architecture were used.Those 3 blocks are namely inception module and CBAM seperated by a convolution block with 3 layers. The overall layer structure of Custom Network is shown in Table 3.1

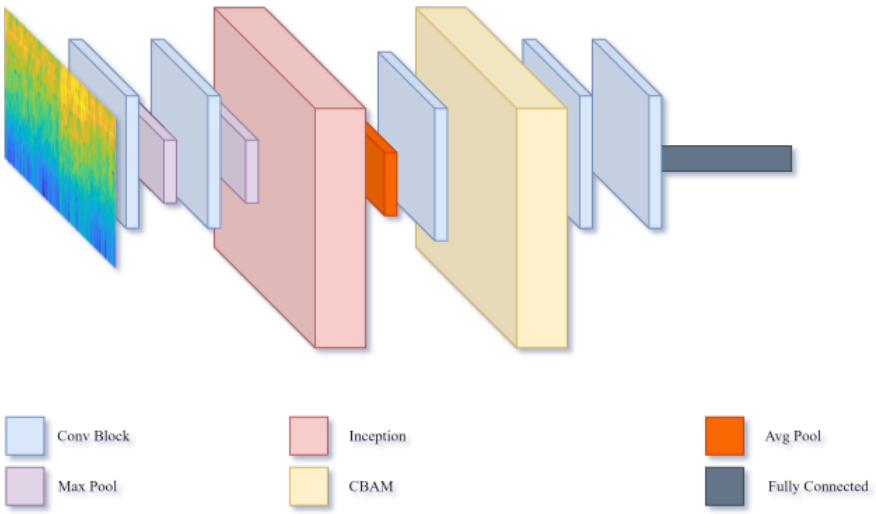


Figure 3.13: Custom Net Architecture

Only one Fully connected layer was used and avoided sigmoid activation function at the end since the loss function used in this Custom Network is "categorical cross entropy" (Categorical Cross entropy applies softmax by default, Reapplying softmax after Fully Connected Layer stalled the training).

Table 3.1: Layer descriptions

Layer (type)	Output Shape	Param
Conv2d-1	[-1, 64, 112, 112]	9,472
BatchNorm2d-2	[-1, 64, 112, 112]	128
ReLU-3	[-1, 64, 112, 112]	0
Conv_Block-4	[-1, 64, 112, 112]	0
MaxPool2d-5	[-1, 64, 56, 56]	0
Conv2d-6	[-1, 192, 56, 56]	110,784
BatchNorm2d-7	[-1, 192, 56, 56]	384
ReLU-8	[-1, 192, 56, 56]	0
Conv_Block-9	[-1, 192, 56, 56]	0
MaxPool2d-10	[-1, 192, 28, 28]	0
Conv2d-11	[-1, 64, 28, 28]	12,352
BatchNorm2d-12	[-1, 64, 28, 28]	128
ReLU-13	[-1, 64, 28, 28]	0
Conv_Block-14	[-1, 64, 28, 28]	0
Conv2d-15	[-1, 96, 28, 28]	18,528
BatchNorm2d-16	[-1, 96, 28, 28]	192
ReLU-17	[-1, 96, 28, 28]	0
Conv_Block-18	[-1, 96, 28, 28]	0
Conv2d-19	[-1, 128, 28, 28]	110,720
BatchNorm2d-20	[-1, 128, 28, 28]	256
ReLU-21	[-1, 128, 28, 28]	0
Conv_Block-22	[-1, 128, 28, 28]	0
Conv2d-23	[-1, 16, 28, 28]	3,088
BatchNorm2d-24	[-1, 16, 28, 28]	32
ReLU-25	[-1, 16, 28, 28]	0
Conv_Block-26	[-1, 16, 28, 28]	0
Conv2d-27	[-1, 32, 28, 28]	12,832
BatchNorm2d-28	[-1, 32, 28, 28]	64
ReLU-29	[-1, 32, 28, 28]	0
Conv_Block-30	[-1, 32, 28, 28]	0
MaxPool2d-31	[-1, 192, 28, 28]	0
Conv2d-32	[-1, 32, 28, 28]	6,176
BatchNorm2d-33	[-1, 32, 28, 28]	64
ReLU-34	[-1, 32, 28, 28]	0
Conv_Block-35	[-1, 32, 28, 28]	0
Inception_block-36	[-1, 256, 28, 28]	0
AvgPool2d-37	[-1, 256, 22, 22]	0
Conv2d-38	[-1, 64, 22, 22]	147,520
AdaptiveAvgPool2d-39	[-1, 64, 1, 1]	0
Linear-40	[-1, 8]	512
ReLU-41	[-1, 8]	0
Linear-42	[-1, 64]	512
AdaptiveMaxPool2d-43	[-1, 64, 1, 1]	0
Linear-44	[-1, 8]	512
ReLU-45	[-1, 8]	0
Linear-46	[-1, 64]	512
Sigmoid-47	[-1, 64]	0
channel_attention_module-48	[-1, 64, 22, 22]	0
Conv2d-49	[-1, 1, 22, 22]	98
Sigmoid-50	[-1, 1, 22, 22]	0
spatial_attention_module-51	[-1, 64, 22, 22]	0
cbam-52	[-1, 64, 22, 22]	0
Conv2d-53	[-1, 34, 22, 22]	19,618
BatchNorm2d-54	[-1, 34, 22, 22]	68
ReLU-55	[-1, 34, 22, 22]	0
Conv_Block-56	[-1, 34, 22, 22]	0
Conv2d-57	[-1, 17, 22, 22]	5,219
BatchNorm2d-58	[-1, 17, 22, 22]	34
ReLU-59	[-1, 17, 22, 22]	0
Conv_Block-60	[-1, 17, 22, 22]	0
Linear-61	[-1, 17]	139,893

Eventhough it is a common practice in many established networks like VGG,AlexNet to add multiple Fully Connected layers, we limited it to one. The main reason for this is to reduce the number of trainable parameters and speed up training.It doesnt mean we didn't experiment with more FC layers. Once more FC s are added training was taking forever to converge it was painfully slow. Table 3.1 Shows the architecture summary of the Custom Network. In Figure 3.13 block diagram of the same is depicted.

Learning Rate

Learning Rate is a crucial parameter if set too high the model will bounce off the sides of minimum.If too low Training would be very slow and resource intensive.Even if we are able to cope with lower training speed still there chance of being stuck in local minimum.So to adjust training rate during the training was found necessary.It was done in 3 phases.Training was done with a constant learning rate of 0.001 initially and found out that after first 3,4 epoch accuracy started declining(accuracy was increasing steadly at a high rate) In first phase it was set to reduce the learning rate by 1/10 th after 3rd epoch.Its accuracy graph also stalled and tumbled after touching a new high(better than previous) In the second phase Learning rate was reduced 1/10 th by looking at loss score.If loss was less than 0.4 reduce the learning rate else keep the earlier learning rate. It gave opportunity to escape from local minima and also at the same time get a good consistent time bound training.

Train Validation Splitting

Deciding train-test split ratio is usually an under rated step.Initially trained with 80:20 split which yielded near 90 percent accuracy for training data but performed not so well in validation data(which was not shuffled).70:30 split made less progress.So final model was trained at 75:25 split ratio.

Preprocessing Transforms

Image resizing was done before hand to increase the speed during training,Other transforms like normalizing with mean and standerd deviation was avoided in later stages since it was'nt helping.No augmentations were done as it would corrupt model to do augmentation on mel spectral images.All images were converted to tensors and fed in batches of 128 using generator.

3.5.6 Convolution Block

Relu Convolutional 2D block Batch Normalization Batch Normalization is known to speed up training (reasons are still disputed over), Batch normalization works by normalizing the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

Relu Activation function is used before taking input from previous convolution layer.Other newer activations like Gelu which promised to introduce non linearity were tested during this work and failed.The Structure of this block can be seen in Table. 3.2.

Table 3.2: Convolutional Block

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 192, 222, 222]	5,376
BatchNorm2d-2	[-1, 192, 222, 222]	384
ReLU-3	[-1, 192, 222, 222]	0
Total params:		**5,760**

3.5.7 Convolutional Block Attention Module

Convolutional Block Attention Module a.k.a CBAM [15] is the concatenation of 2 modules namely Channel Atttention Module and Spatial Attention Module as shown in Figure 3.14 [15]. In the Custom Network CBAM was used towards the middle part of network. Studies suggest that it gives better performance when placed in between.We also verified this claim partly by placing CBAM at the begining of network and the studies were proved right.

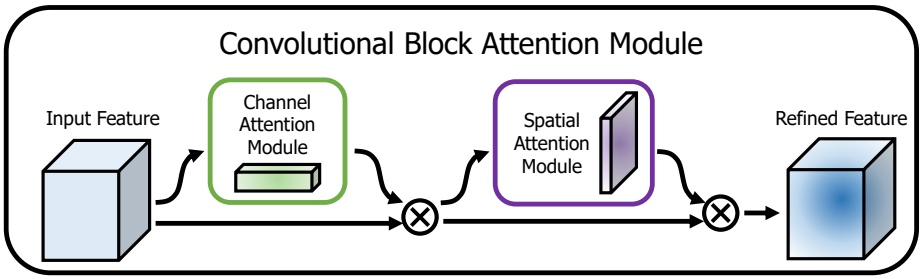


Figure 3.14: CBAM

Channel Attention Module

A Channel Attention Module Figure. 3.15 [15] is a module for channel-based attention in convolutional neural networks. We produce a channel attention map by exploiting the inter-channel relationship of features. As each channel of a feature map is considered as a feature detector, channel attention focuses on 'what' is meaningful given an input image. To compute the channel attention efficiently, we squeeze the spatial dimension of the input feature map.

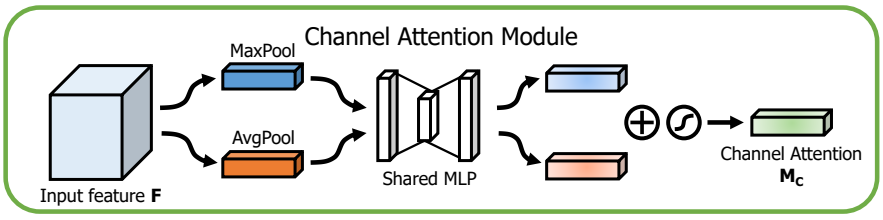


Figure 3.15: Channel Attention Module Structure

Spatial Attention Module

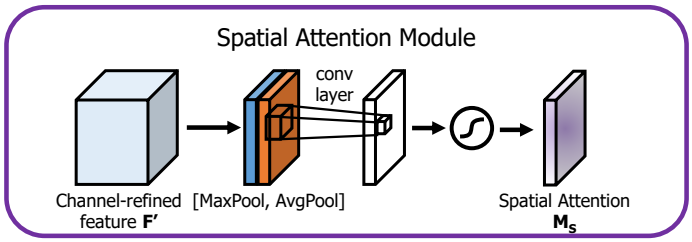
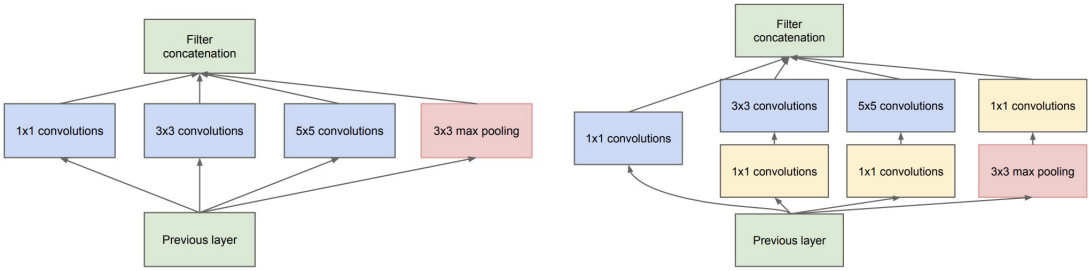


Figure 3.16: Spatial Attention Module

A Spatial Attention Module (whose structure is shown in Figure. 3.16 [15])is a module for spatial attention in convolutional neural networks. It generates a spatial attention map by utilizing the inter-spatial relationship of features. Different from the channel attention, the spatial attention focuses on where is an informative part, which is complementary to the channel attention. To compute the spatial attention, we first apply average-pooling and max-pooling operations along the channel axis and concatenate them to generate an efficient feature descriptor.

3.5.8 Inception Module



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 3.17: Inception Block Architecture

Inception module is used at the begining of network,Just after the initial Convolutional block. The advanced version of inception module [16] shown in Figure 3.17 b [16] was used for the design of custom network. The Inception module is a building block in many deep learning architectures, particularly popularized by the Inception network (GoogLeNet). It consists of multiple parallel convolutional layers of different filter sizes and pooling operations, aimed at capturing features at multiple scales within the same network layer. This design enables efficient and effective feature extraction, facilitating the learning of both local and global patterns in the input data.Different sized kernels could effectively absorb most of the features.

Chapter 4

Results and Discussion

4.1 Experiments on data pre-processing task

The preprocessing steps are carefully carried out for dataset and a complete matrix of whole data was obtained. The code was executed successfully to generate a comprehensive dataset, encapsulated in a cell of a complete set. This set, represented as a matrix of dimensions (71680 rows) by (32 columns), signifies the culmination of the data rearrangement process. Each row within the matrix corresponds to a specific trial of a gesture, with a total of 7 trials captured for each gesture. The temporal extent of each gesture is 5 seconds, resulting in 35 seconds of data for each trial at a sampling rate of 2048 samples per second. Each column in this new matrix encapsulates the readings of a single electrode across the multiple trials for a particular gesture. At the end after preprocessing we obtained dataset with the structure as given in Figure. 4.1.

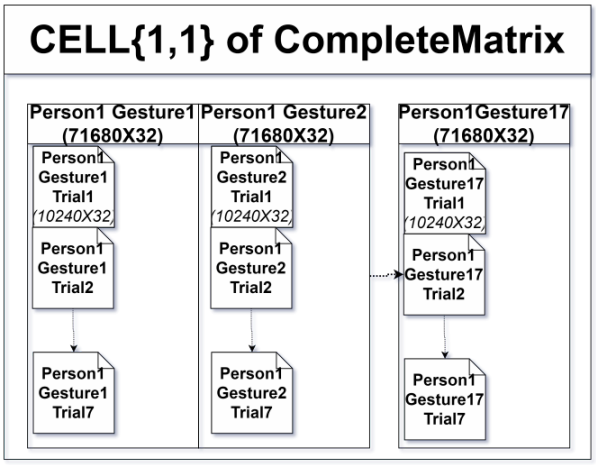


Figure 4.1: Dataset Structure After Preprocessing

4.2 Hardware

Leveraging a robust computational environment, signal processing and machine learning tasks were performed using MATLAB on a high-performance system equipped with an RTX

3050 graphics card, 16 GB of RAM, a 512 GB SSD, and an Intel Core i5 12th generation processor. The powerful GPU capabilities of the RTX 3050, coupled with ample system memory, enabled efficient parallel processing and accelerated computations, particularly beneficial for intensive tasks such as real-time signal processing. The inclusion of a high-speed SSD contributed to swift data access and retrieval, optimizing overall workflow efficiency. The Intel i5 12th gen processor provided the computational muscle required for intricate MATLAB algorithms, ensuring seamless execution of tasks related to data analysis, feature extraction, and machine learning model training. This well-balanced hardware configuration created a synergistic platform, empowering exploration of complex data sets, real-time simulations, and development of sophisticated algorithms with a high level of computational efficiency and responsiveness.

4.3 Analysis of Results Obtained(Machine Learning)

4.3.1 Complete set

After the rearrangement of samples a complete set matrix was obtained which essentially merged all .mat files to a single large file as shown in Figure 4.2 in an ordered and easily accessible way.

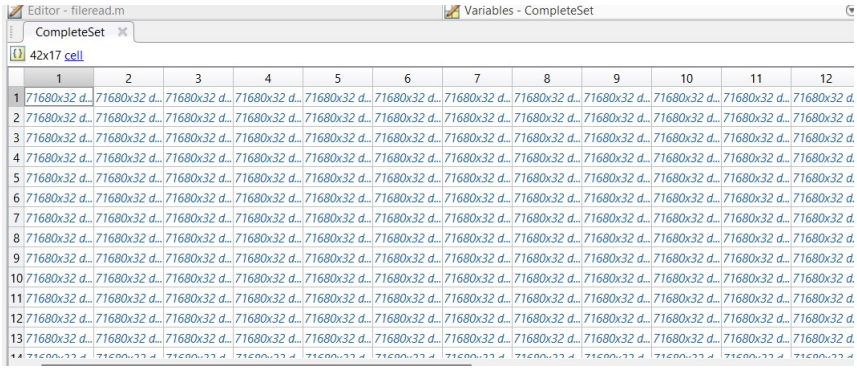


Figure 4.2: Complete Set

4.3.2 KNN Model

After training the KNN model with different set of parameters and the best ones were found out and are listed in Table 4.1.

Table 4.1: Model Hyperparameters: Weighted KNN

Hyperparameter	**Value**
Preset	Weighted KNN
Number of Neighbors	10
Distance Metric	Euclidean
Distance Weight	Squared Inverse
Standardize Data	Yes

Table 4.2: KNN Model Performance Metrics

Metric	Value
Accuracy (Validation)	79.1%
Total Cost (Validation)	990
Prediction Speed	10000 obs/sec
Training Time	14.885 seconds
Model Size	1007 KB

Model Summary

The Weighted KNN model, leveraging 24 features, demonstrates robust performance across various metrics. Notably, the model achieves an accuracy of 79.1 percent on the validation set as shown in Table 4.2 , accompanied by a total cost of 990. With a prediction speed of 10,000 observations per second, the model exhibits rapid inference capabilities. The training process is completed efficiently in 14.885 seconds. Moreover, the compact size of the model, standing at 1007 KB, is favorable for storage and computational efficiency. The model's hyperparameters are set for weighted KNN with 10 neighbors, utilizing the Euclidean distance metric and squared inverse distance weighting. Standardization of the input data is applied to enhance model performance. Confudion matrix obtained during validation is shown in Figure. 4.3

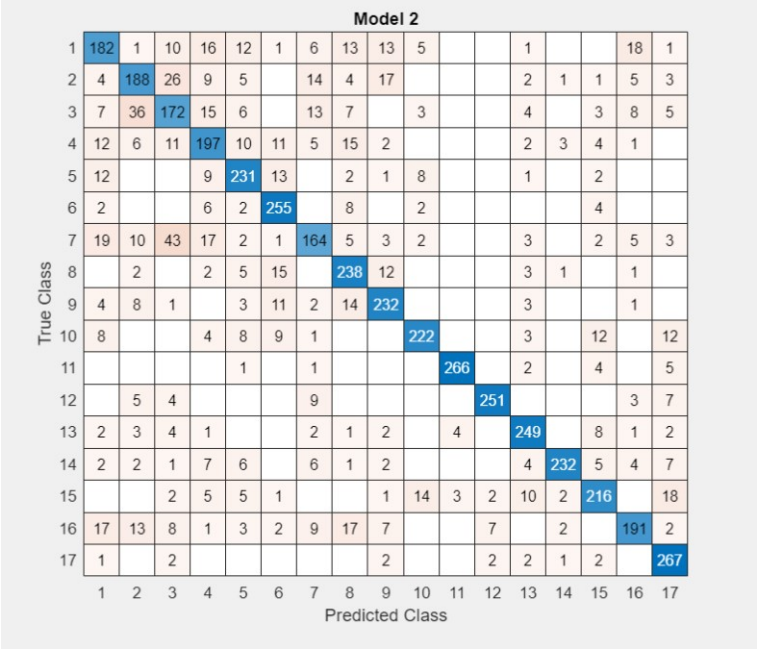


Figure 4.3: Confusion Matrix of KNN Model

4.3.3 SVM Model

Optimized Hyperparameters

The hyperparameters for the optimized model are as follows:

- **Kernel Function:** Quadratic
- **Box Constraint Level:** 0.19447
- **Multi-class Coding:** one vs one
- **Standardized Data:** Yes

These hyperparameter values were determined through optimization to enhance the performance of the model.

Hyperparameter Search Range

Table 4.3: Hyperparameter Search Range

Hyperparameter	Search Range
Multi-class Coding	One-vs-All, one vs one
Box Constraint Level	0.001–1000
Kernel Scale	0.001–1000
Kernel Function	Gaussian, Linear, Quadratic, Cubic
Standardized Data	True, False

In the hyperparameter search, different options were considered for each parameter (Table 4.3) to find the optimal combination for high performance model.

Table 4.4: SVM Model Performance Metrics

Metric	Value
Accuracy (Validation)	84.0%
Total Cost (Validation)	760
Prediction Speed	2200 obs/sec
Training Time	9669.4 sec
Model Size	3MB
Kernel Scale	1

Model Summary

The SVM model discussed above exhibits strong performance, achieving an accuracy of 84.0 percent on the validation set with a total cost of 760 as shown in Table 4.4 Notably, the model demonstrates efficient prediction speed, handling 2200 observations per second. The training process took 9669.4 seconds, resulting in a compact model size of 3MB. Confusion Matrix

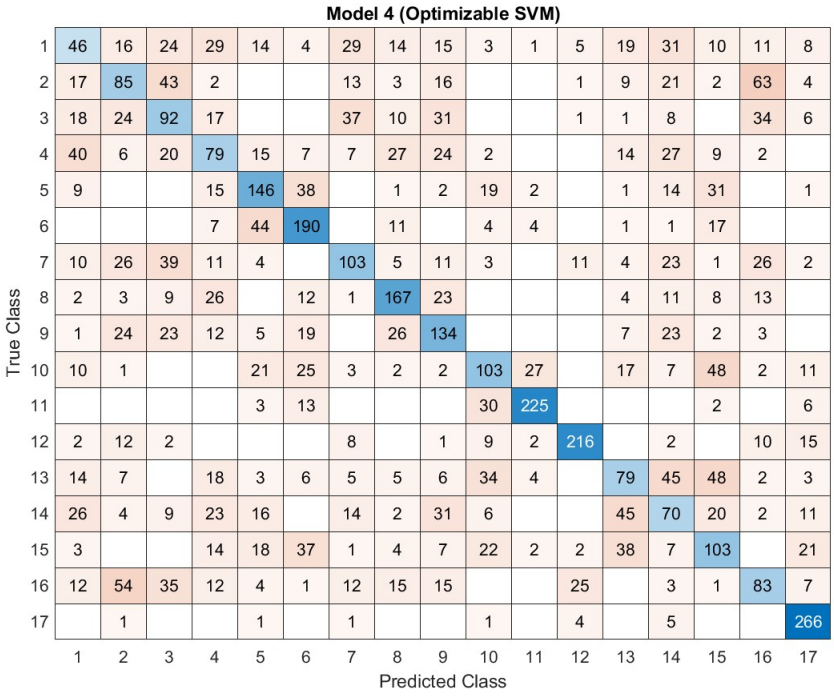


Figure 4.4: Confusion Matrix SVM Model

after validation is shown in Figure. 4.4. Through careful hyperparameter tuning, the model's kernel function was optimized to Quadratic, with a box constraint level of 0.19447 and a multiclass coding strategy of one vs one. The use of standardized data further contributed to the model's effectiveness. The hyperparameter search encompassed diverse ranges, exploring multiclass coding methods (One-vs-All, one vs one), box constraint levels (0.001–1000), kernel scales (0.001–1000), various kernel functions (Gaussian, Linear, Quadratic, Cubic), and the consideration of standardized versus non-standardized data. This comprehensive exploration aimed to identify the most suitable hyperparameter combination, ultimately enhancing the SVM model's accuracy and predictive capabilities.

4.3.4 Comparison of models

In comparison between the SVM and KNN models, the SVM model outperforms in several key aspects as is evident from Table 4.5. The SVM model achieved a higher accuracy of 84.0 percent on the validation set, exceeding the 79.1 percent accuracy achieved by the KNN model. Additionally, the SVM model demonstrated a competitive prediction speed of 2200 observations per second, while the KNN model exhibited a speed of 10000 observations per second. Although the SVM model had a longer training time of 9669.4 seconds compared to the KNN model's 14.885 seconds, the SVM model's superior accuracy suggests that the additional training time may be justified. Furthermore, the SVM model showcased a more compact model size of 3MB compared to the KNN model's 1007 KB. Overall, considering the higher accuracy and a reasonably competitive prediction speed, the SVM model emerges

as the more favorable choice for the given application.

Table 4.5: Comparison Between SVM and KNN Models

Model Metric	**SVM**	**KNN**
Accuracy (Validation)	84.0%	79.1%
Total Cost (Validation)	760	990
Prediction Speed	2200 obs/sec	10000 obs/sec
Training Time	9669.4 seconds	14.885 seconds
Model Size	3MB	1007 KB
Kernel Function (SVM)	Quadratic	-
Number of Neighbors (KNN)	-	10
Distance Metric (KNN)	-	Euclidean
Distance Weight (KNN)	-	Squared Inverse
Standardized Data	Yes	Yes

4.4 Results of Anatomical Study on Forearm

(a) Flexor digitorum superficialis (b) Flexor Carpi Ulnaris

Figure 4.5: Flexor Compartment

Flexor digitorum superficialis

The flexor digitorum superficialis Figure 4.5a is a forearm muscle originating from the medial epicondyle of the humerus and the coronoid process of the ulna. It travels down the forearm and inserts into the middle phalanges of the fingers. Its primary function is to flex the fingers at the proximal interphalangeal joints, assisting in gripping and grasping movements. Assist

in movements of wrist, hand and fingers Flexion of middle and proximal phalanges of digits 2-5 Flexion of hand at wrist joint.

Flexor carpi ulnaris

The flexor carpi ulnaris Figure 4.5b is a forearm muscle with origins at the medial epicondyle of the humerus and the posterior border of the ulna. Its main functions include wrist flexion and ulnar deviation, aiding in movements that bring the palm closer to the forearm and towards the ulnar side of the hand. Innervated by the ulnar nerve, this muscle is essential for stabilizing the wrist and contributing to various hand and wrist movements like Flexing Wrist Abducting the hand.

(a) Flexor Carpi Radialis (b) Extensor Digitorum

Figure 4.6: Muscles in Focus

Flexor Carpi Radialis

The flexor carpi radialis Figure 4.6a is a forearm muscle that originates from the medial epicondyle of the humerus. It runs down the forearm and inserts onto the base of the second metacarpal bone. Its primary function is wrist flexion and radial deviation, assisting in movements that bring the palm closer to the forearm and towards the radial side of the hand.

Extensor Digitorum

Assist in movements of wrist, hand and fingers Extension of joints of digits 2-5 Extension of hand at wrist joint. The extensor digitorum Figure. 4.6b is a forearm muscle originating from the lateral epicondyle of the humerus. It extends down the forearm and inserts into the middle and distal phalanges of the fingers. Its main function is to extend the fingers at the metacarpophalangeal, proximal interphalangeal, and distal interphalangeal joints.

4.4.1 Conclusion of Anatomical Study

The muscle groups which are more important for making the gestures in question where identified as Flexor carpi radialis,Extensor Digitorum, Flexor carpi ulnaris and Flexor digitorum superficialis.Comparing their locations and insertions with the positioning of electrodes it can be concluded that electrodes numbered 1,2,4,6 are more relevant for this study.

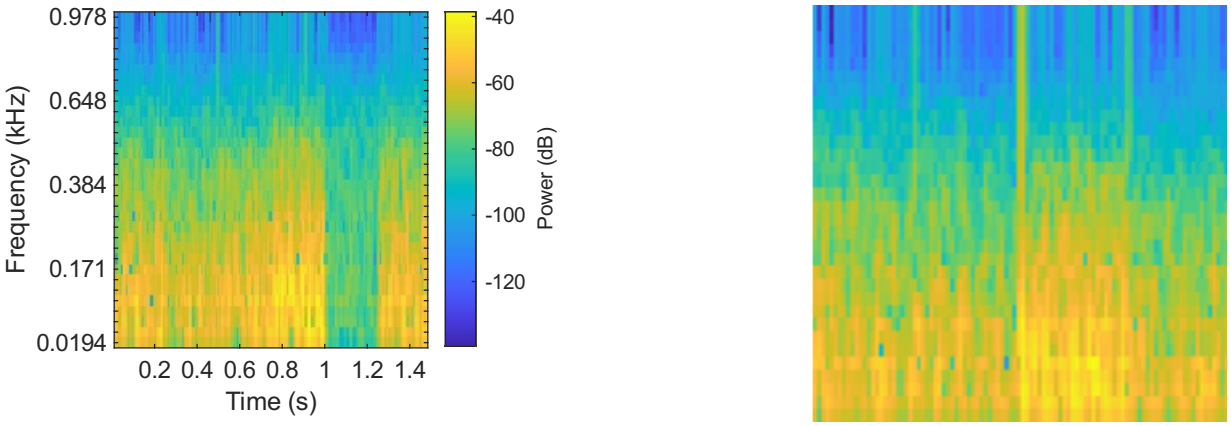
4.5 Results of Statistical Study on Features

After obtaining the correlation matrix (shown in Figure 3.9) of various features as discussed earlier in methodology ,by manual observation it was identified that electrodes 1,2,4,6 are more independent.

4.6 Analysis of Results Obtained(Deep Learning)

4.6.1 Dataset Creation

GrabMayo dataset structure Dataset to be used for classification of gestures was spectral images. Spectral image of each window (279 mel spectrum image for a single persons single gesture) were created and stored to 17 folders with corresponding gesture name. Sample figure of 2 melspectrum images from 2 windows are shown below.(Figure 4.7a and Figure. 4.7b) Figure 4.7a shows the scale and axis values of mel spectrum image. When used for classification only the cropped part is considered,axis are turned off and empty white space around the spectral image were cropped along the boundary(Figure. 4.7b).



(a) person 1 gesture 1 window 1

(b) person 1 gesture 6 window 23

Figure 4.7: Spectral Image

4.6.2 Results of Alex Net

AlexNet on applying Transfer learning produced accuracy of 84 percent after 30 epoch training. During most part of training validation accuracy(black) is more than that of training accuracy. This may seem awkward but since during the validation the model avoids dropouts and batch normalization this is acceptable. It cannot be treated as a sign of over fitting.

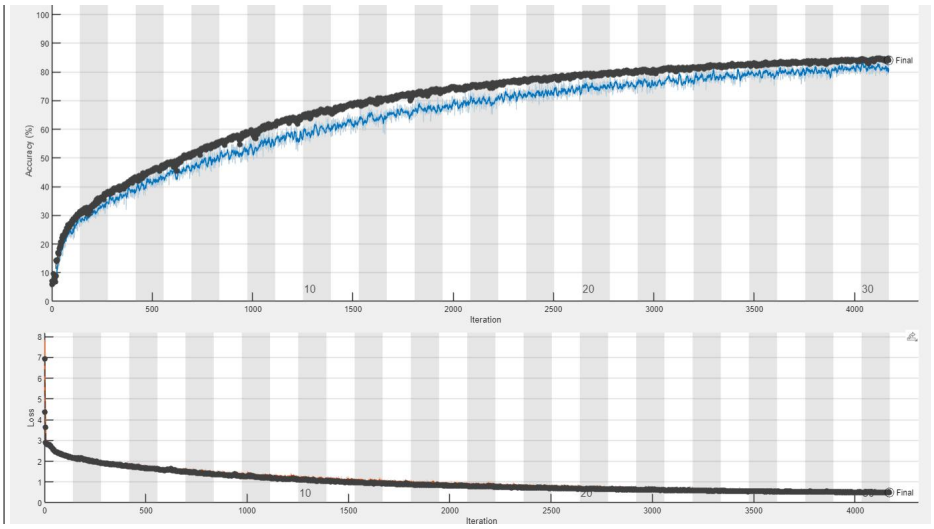


Figure 4.8: AlexNet training validation plot

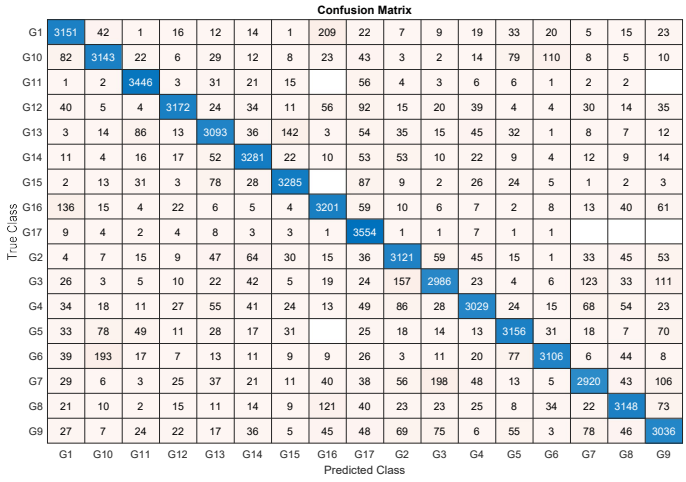


Figure 4.9: AlexNet Confusion Matrix

It appears that the model performs well on some classes (e.g., G8, G16) but not as well on others (e.g., G13, G15). For example, looking at row G13 (gestures of class G13), the model predicted a significant portion of them as G14, G15, and G18, indicating these classes might be confused with G13. Overall, the confusion matrix given in Figure. 4.9 provides valuable insights into how well the AlexNet model is classifying gestures and helps identify areas for potential improvement.

4.6.3 Results of VGG16

From the training plot given in Figure 4.10 it is evident that VGG16 performs far better than its image net predecessor AlexNet. Usually one would measure the size of the network in number of parameters (i.e. weights) and number of layers. VGG-16 has 138 million parameters which is notably large.

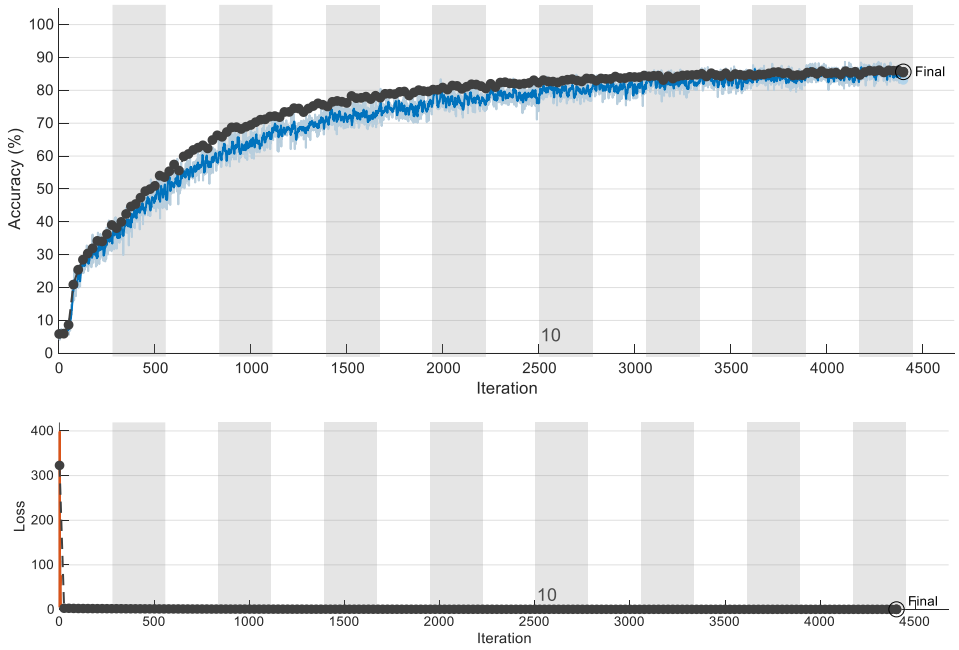


Figure 4.10: VGG16 Train Validation

It reached an accuracy of 86.3% at 16 th epoch.Training was stopped at 16 th epoch as the model converged at that point no possibility for further improvement in accuracy was visible.

Confusion matrix indicates better performance than AlexNet and on certain classes of gestures like gesture 17 misclassification rate was very meagre. Confusion metrics of classification are given below in Figure. 4.11.

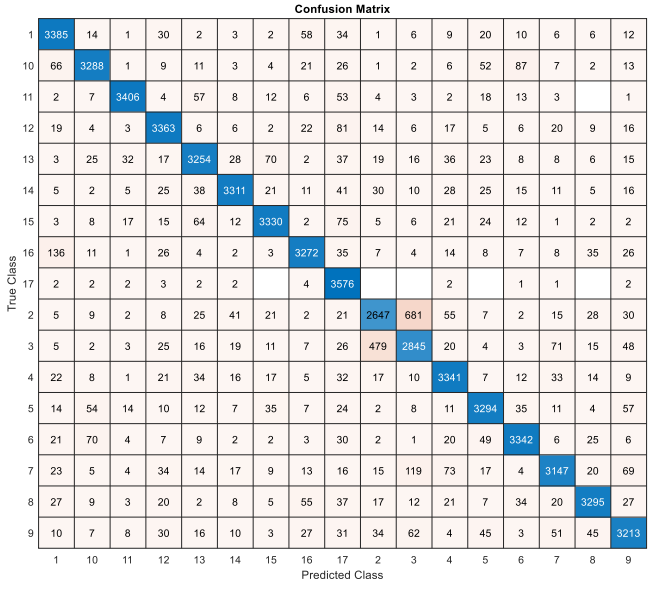


Figure 4.11: VGG16 Confusion Matrix

4.6.4 Results of ResNet50

The ResNet50 model was trained to classify 17 gesture melspectrograms using a learning rate of 0.0001. The training process took approximately 7 days in Nvidia RTX3060 64 GB RAM machine .

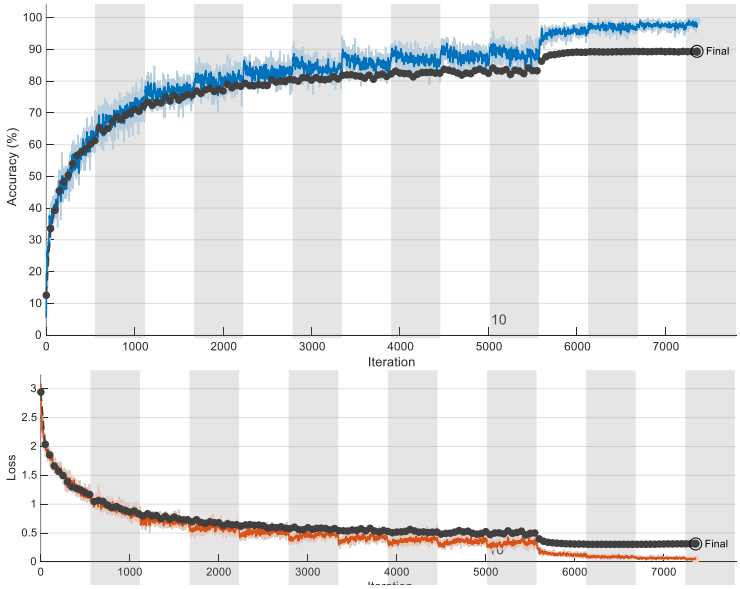


Figure 4.12: RESNT50 Train Validation

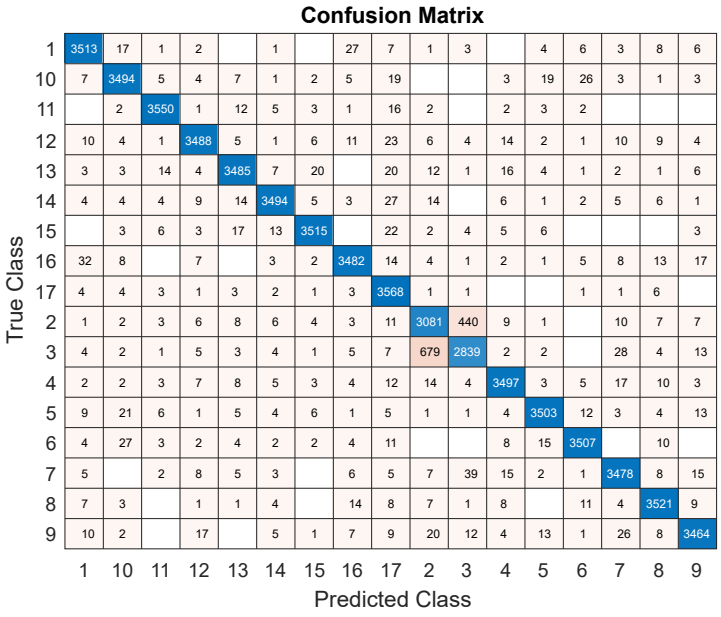


Figure 4.13: RESNET50 Confusion Matrix

Validation was performed every 50 th iteration to monitor model's performance on unseen data. Training continued for 16 epoch when the validation accuracy started to converge. The

final model achieved satisfactory performance on the classification task, as detailed in the following section.ResNet50's train validation plot after transfer learning is shown in Figure. 4.12 The ResNet50 Model reached an accuracy of 89.38 once it reached 16 th epoch. Training was stopped at 16 th epoch as the model converged at that point no possibility for further improvement in accuracy was visible.

The confusion Matrix of 17 Gestures are shown in Figure. 4.13 Gestures 2 and 3 shows high miss classification rate. For other gestures this model works fine.

4.6.5 Results of Xception

From the training and validation plot of Xception model we can infer that the model overfitted with the data producing 100 percent accuracy with 84.5 percent validation accuracy. Model was trained for 30 epochs and with a batch size of 128 images. Learning rate was set at 0.0001. Training validation plot of Xception is shown in Figure. 4.14

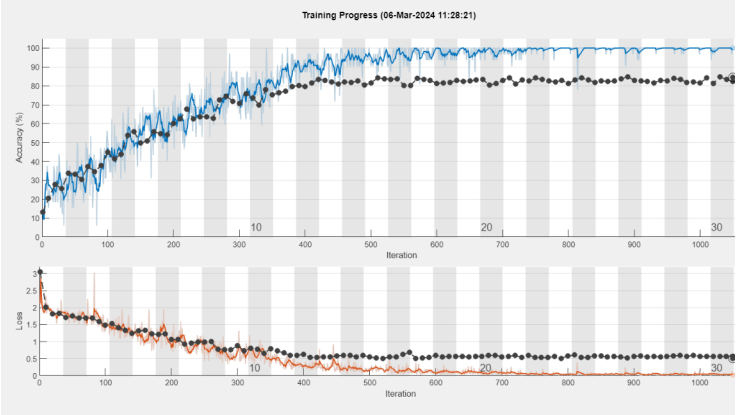


Figure 4.14: Xception Training Validation Plot

4.6.6 Results of Mobile Net

A model developed using MobileNet V2 was trained during this work, which is known for its stability. However, it was only able to achieve an accuracy of 35 percent. The validation accuracy plateaued at 35 percent and did not improve, resulting in an ineffective model.

4.6.7 Results of Custom Net

Custom Net was trained using Pytorch in Pycharm IDE. Training took more than 15 Hours. Training was done without any normalization of inputs. Machine used was RTX 4060 with 64 GB RAM.

Training Loss per Batch

Plotting training loss per batch helped in understanding when to adjust learning rate(at which epoch or at which loss value). Figure. 4.16 shows the loss plot.

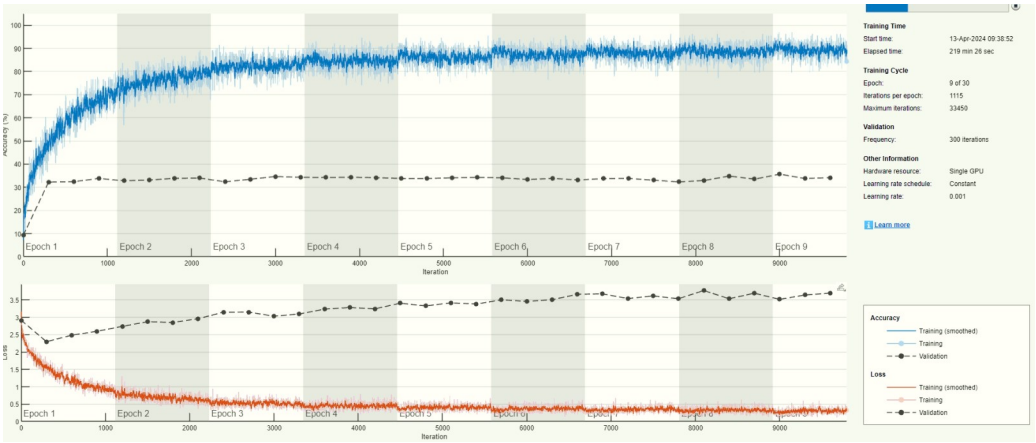


Figure 4.15: MobileNet Training Plot

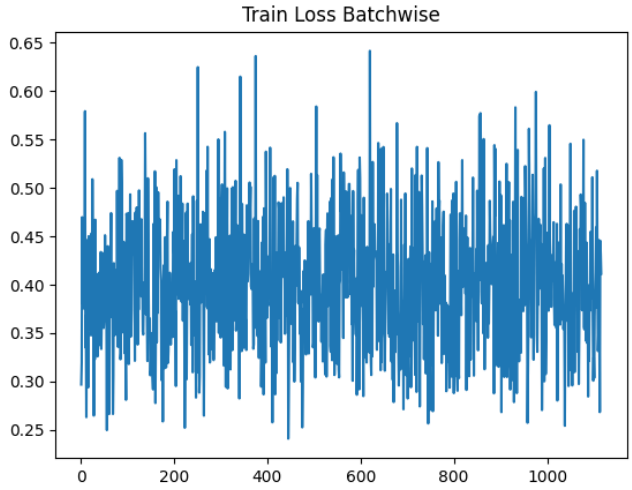


Figure 4.16: Train Loss Batch Wise

Training and validation accuracy

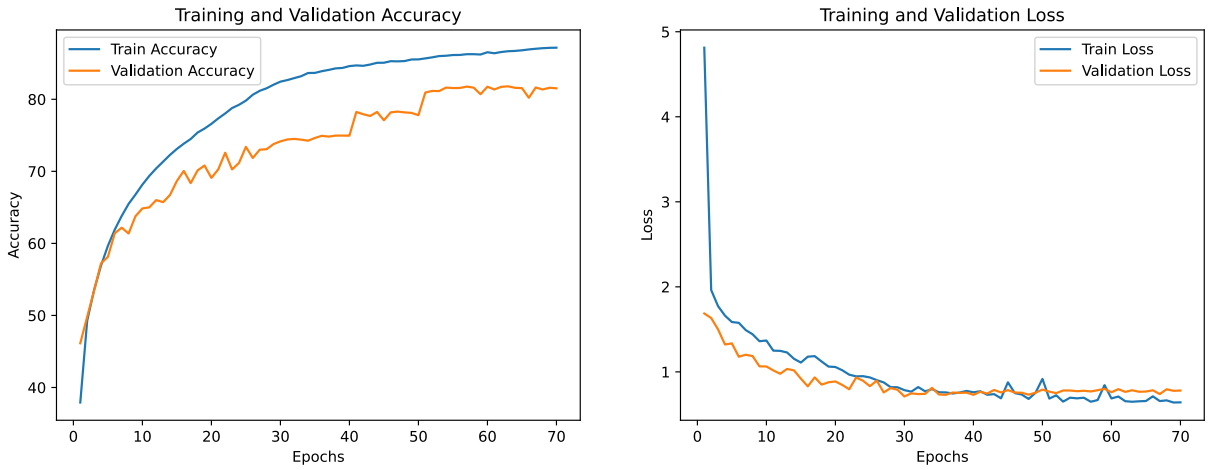
Training and validation accuracy plot for 70 epoch training is shown in Figure. 4.17a Training accuracy reached 87.16 percent while the best validation accuracy was 81.79 percent.

Confusion Matrix

Confusion matrix was formed on a validation set of 25% data. Gesture 14 acquired the most correct number of predictions with just one wrong prediction.

Precision and Recall

One could observe that precision and recall Figure. 4.19a are improving in a similar fashion as that of accuracy. It must be noted that it is the average precision and recall that are used here instead of conventional precision and recall.



(a) Training and Validation Accuracy

(b) Training and Validation Loss

Figure 4.17: Training and Validation Plots

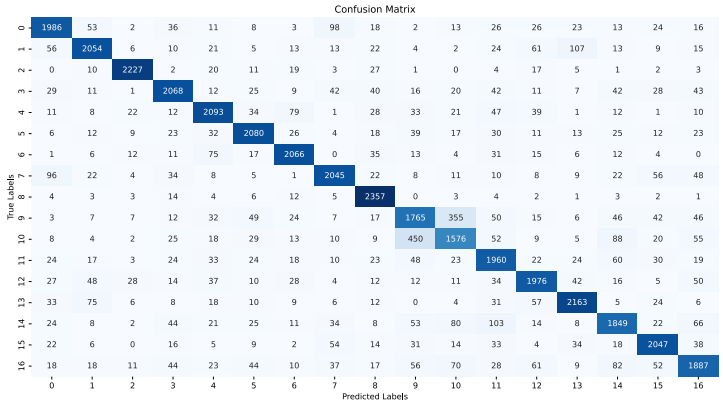


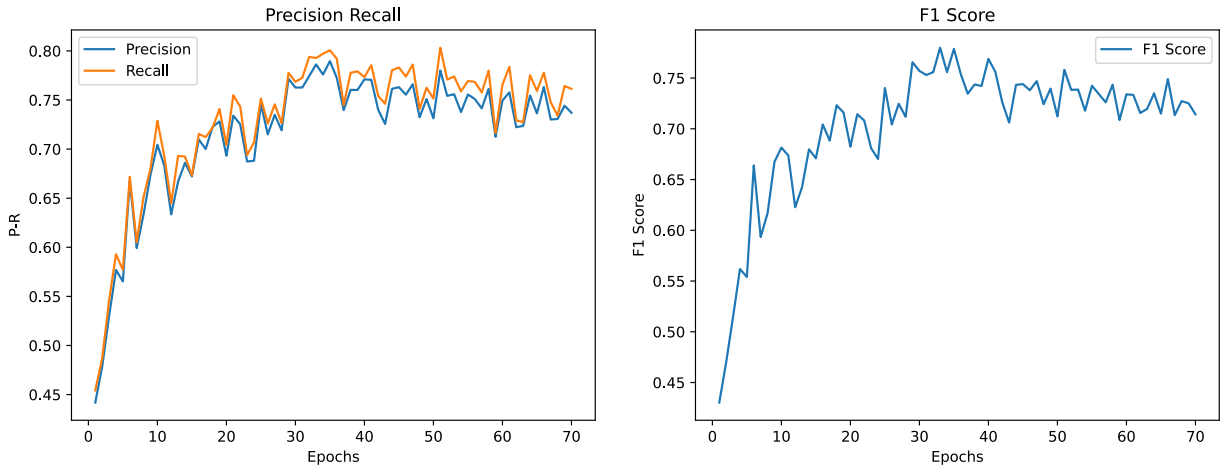
Figure 4.18: Confusion Matrix of Custom Model

F1 Score

F1 score being the harmonic mean of precision and recall exhibit similar variation but with different values.

Latency

Latency was measured using time module in python in a machine with RTX 3050 Graphics card. It was found to take 3-5 ms (on an average it can be treated as 4 ms) for a single image to pass through model and yield output. It may increase or decrease depending on machine specifications. Concurrently pre processing the input signals and converting it to melspectrogram can prevent further delay during real time deployment.



(a) Precision And Recall

(b) F1 Score

Figure 4.19: Evaluation Metrics Plot

4.7 Comparison of Deep Learning Models

It is evident from the Table 4.6 Latency of the Custom Network developed during this work is the lowest(less than 4 ms on an average) even less than the mobilenet which yields only 35% accuracy at 5 milliseconds time. Custom Network with less than 6 lakh parameters is competing well in accuracy with the large deep learning models having millions of parameters. All the latency measures were tested in Nvidia RTX3050 Graphics card and in the environment of pycharm ide.

Table 4.6: Performance Metrics of Different Models

Model	**Params**	**Accuracy(%)**	**Precision(%)**	**Recall(%)**	**F1 Score(%)**	**Latency**
AlexNet	62.3 M	85	84.68	83.44	84.05	10 ms
VGG16	138 M	85.3	87.01	84.28	85.62	15 ms
ResNet50	25.56 M	89.38	89.32	89.23	89.27	30 ms
Custom Net	599698	81	79	78	78.50	4 ms
Mobile Net	3.4 M	35	34	32	32.96	5 ms

Custom Net less than 6 lakh parameters,comparing to AlexNet which has 62 Million parameters Custom Net is definitely a worthy model for spectral image classification(Respecting the fact that AlexNet was trained on 1000 classes). Other models like VGG are far more heavier with billions of parameters. While comparing with small and efficient models like Mobile net which have 3 million parameters we achieved far more accuracy.From the Figure 4.15 we can understand that mobile net started overfitting the data even before 10 epochs. So it cannot be treated as a light weight network with reasonable accuracy. Here we achieved a delicate balance between network size and ability to explain the data. Figure 4.20 shows the graphical representation of comparison.

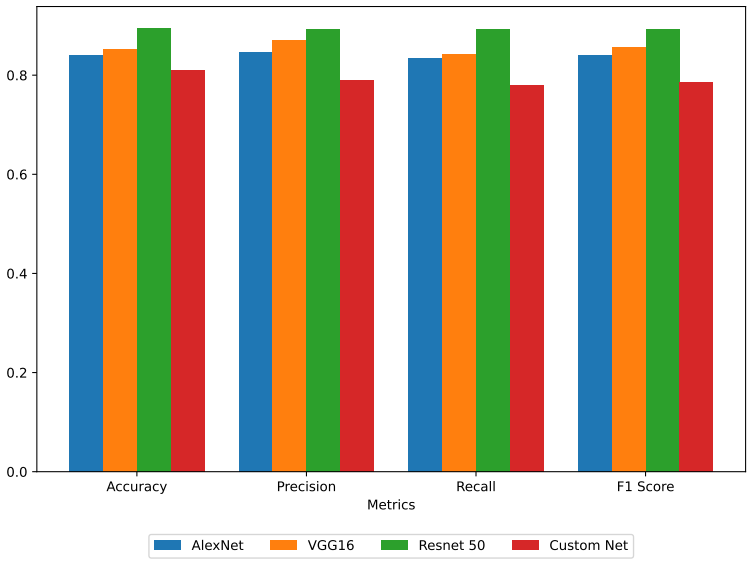


Figure 4.20: Comparison of Evaluation Metrics

4.8 Simulink Implementation

Simulink Model was created using different blocks.The trained model's predicted result(*deep learning image classifier*) was converted to a constant between 1 to 17(*constant block*) and passed to switch case and subsequently to *if action block*. A merge block was added to combine it and take one output action(Multiple actions and sub tasks can also be included). In this example shown in Figure. 4.21 the output action is to display the corresponding gestures image (*video viewer block of computer vision tool box*) Matlab 2024b version was used for building simulink model.

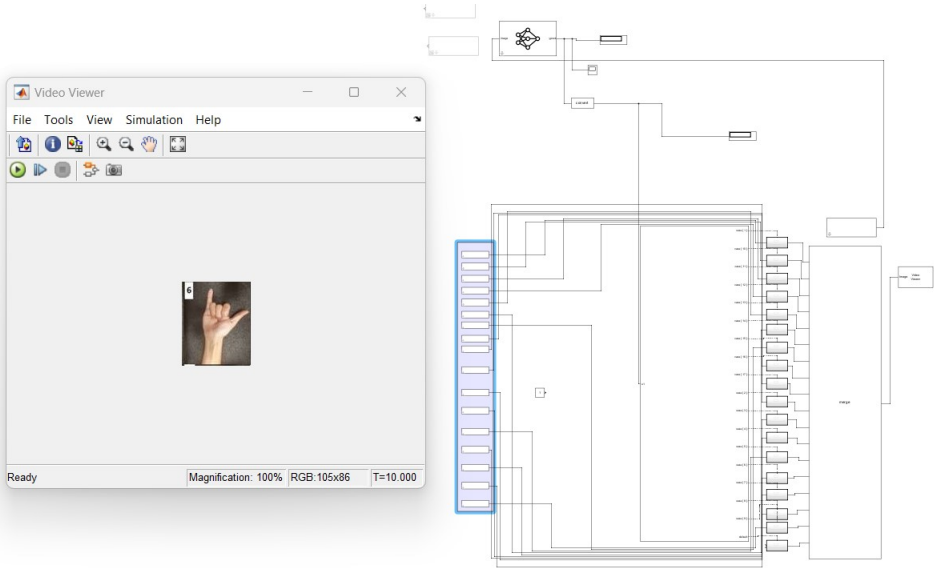


Figure 4.21: Simulink Implimentation

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

Several models were trained using transfer learning techniques and developed a Custom Network during the course of this work. The Custom Network developed during course of this work limited the number of parameters to around 5 lakhs while the various transfer learned models stand at millions of parameters. So a huge reduction in number of parameters were achieved without compromising heavily on accuracy. The developed model was also super fast with a prediction speed of 0.004 seconds on an average which means it can be used for realtime deployment. Many models behave differently on different gestures.We have already observed that VGG 16 mis classified gestures 2 and 3 but for some other gestures combinations it works very well. Other Transfer learned models also exhibit similar patterns. So we can make an ensemble or give priority(weightage) to certain models predictions regarding certain gestures(The gesture combinations they are strong at).

Reducing model size by using various methods like projection, pruning etc. can be tried in future to enhance the performance and delay in processing. Selecting appropriate EMG feature combinations can overcome the impact of the studied disturbances on EMG pattern classification to a certain extent; however, this simple solution is still inadequate. Stabilizing electrode contact locations and developing effective classifier training strategies are suggested to further improve the robustness of EMG based hand gesture recognition.

5.2 Future Scope

Looking ahead, there are several possibilities for future research and development in this field:

- Advanced Filtering Methods: Developing better noise reduction and signal enhancement techniques to improve the quality of raw EMG signals.
- Sports Rehabilitation: Integrate SEMG-based hand gesture recognition into sports rehabilitation programs. Customized rehabilitation exercises, monitored through gestures, can promote targeted muscle activation and recovery.

- Gesture-Controlled Sports Equipment:Explore the integration of SEMG signals for controlling sports equipment, such as gesture-controlled sports gear or wearables for athletes. This can enhance the overall sports experience and performance monitoring.
- Parkinson's Disease and Motor Disorders: Investigate the application of SEMG-based hand gesture recognition for monitoring and assisting individuals with Parkinson's disease and other motor disorders. Customized interventions can be designed based on real-time gesture analysis.
- Telehealth Applications:Implement SEMG-based hand gesture recognition in telehealth applications, allowing healthcare providers to remotely guide patients through exercises and monitor rehabilitation progress.
- Prosthetic Limb Control: Further advance the use of SEMG signals for controlling prosthetic limbs. Develop more intuitive and responsive prosthetic devices that allow users to perform complex hand gestures for enhanced functionality.

References

[1] P. Maceira-Elvira, T. Popa, A.-C. Schmid, F. C. Hummel, Wearable technology in stroke rehabilitation: towards improved diagnosis and treatment of upper-limb motor impairment, Journal of neuroengineering and rehabilitation 16 (2019) 1–18.

[2] A. Van Ommeren, B. Sawaryn, G. Prange-Lasonder, J. Buurke, J. Rietman, P. Veltink, Detection of the intention to grasp during reaching in stroke using inertial sensing, IEEE transactions on neural systems and rehabilitation engineering 27 (10) (2019) 2128–2134.

[3] X. Song, S. Chen, J. Jia, P. B. Shull, Cellphone-based automated fugl-meyer assessment to evaluate upper extremity motor function after stroke, IEEE Transactions on Neural Systems and Rehabilitation Engineering 27 (10) (2019) 2186–2195.

[4] W. Guo, X. Sheng, H. Liu, X. Zhu, Development of a multi-channel compact-size wireless hybrid semg/nirs sensor system for prosthetic manipulation, IEEE Sensors Journal 16 (2) (2015) 447–456.

[5] A. Ninu, S. Dosen, S. Muceli, F. Rattay, H. Dietl, D. Farina, Closed-loop control of grasping with a myoelectric hand prosthesis: Which are the relevant feedback variables for force control?, IEEE transactions on neural systems and rehabilitation engineering 22 (5) (2014) 1041–1052.

[6] A. Fougner, Ø. Stavdahl, P. J. Kyberd, Y. G. Losier, P. A. Parker, Control of upper limb prostheses: Terminology and proportional myoelectric control—a review, IEEE Transactions on neural systems and rehabilitation engineering 20 (5) (2012) 663–677.

[7] M. F. Qureshi, Z. Mushtaq, M. Z. ur Rehman, E. N. Kamavuako, Spectral image-based multiday surface electromyography classification of hand motions using cnn for human–computer interaction, IEEE Sensors Journal 22 (21) (2022) 20676–20683.

[8] M. A. Ahmed, B. B. Zaidan, A. A. Zaidan, M. M. Salih, M. M. B. Lakulu, A review on systems-based sensory gloves for sign language recognition state of the art between 2007 and 2017, Sensors 18 (7) (2018) 2208.

[9] A. Rashid, O. Hasan, Wearable technologies for hand joints monitoring for rehabilitation: A survey, Microelectronics Journal 88 (2019) 173–183.

[10] A. Pradhan, J. He, N. Jiang, Multi-day dataset of forearm and wrist electromyogram for hand gesture recognition and biometrics, Scientific data 9 (1) (2022) 733.

[11] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems 25 (2012).

[12] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[13] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[14] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

[15] S. Woo, J. Park, J.-Y. Lee, I. S. Kweon, Cbam: Convolutional block attention module, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 3–19.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.