Gesture Recognition for Sign Language using Machine Learning

Pradeep Kumar Gupta, Keerthana Ajith and Nayantara Varadharajan
Computer Science Engineering

AMRITA VISHWA VIDYAPEETHAM
BANGALORE, INDIA

Corresponding Author: Nayantara Varadharajan

Abstract -
Sign language serves as a vital form of communication for millions of individuals worldwide who are deaf or hard of hearing. However, the significant communication gap between sign language users and the broader community persists, hindering effective interaction and inclusivity. This project endeavors to address this challenge by developing an advanced gesture recognition model capable of accurately identifying and interpreting sign language gestures from video input. This project focuses on developing a gesture recognition model for sign language, aimed at enhancing communication for individuals who rely on sign language. Leveraging machine learning and computer vision, the system interprets sign language gestures from video input in real-time. With a user-friendly interface, it translates these gestures into text or spoken language, breaking down communication barriers and promoting inclusivity. This technology holds promise for a more connected and inclusive society, benefiting both sign language users and the broader community.In summary, the "Gesture Recognition for Sign Language" project aspires to create an innovative solution that not only empowers sign language users but also promotes a world where effective communication knows no bounds. Through cutting-edge technology, this project endeavors to bridge communication gaps and foster greater inclusivity for all.

1. INTRODUCTION

The World Health Organisation (WHO) estimates that 466 million individuals worldwide[1] suffer from a hearing loss that is incapacitating. These astounding figures include people of all ages, from little children to the elderly. The major form of communication for millions of people throughout the world, including those who are Deaf, Hard of Hearing, and have Speech and Language Disorders, is sign language.

[1] [1]B. O. Olusanya, A. C. Davis, and H. J. Hoffman, “Hearing loss grades and the <I>international classification of functioning, disability and health</i>,” *Bulletin of the World Health Organization*, 01-Oct-2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6796665/. [Accessed: 10-Sep-2023]

According to the International Labour Organisation (ILO), just 23.5% of people with impairments are employed globally. Recognition of sign language has the potential to expand career prospects and foster financial independence by facilitating successful communication in the workplace.

Beyond just being useful, understanding sign language gestures is a message of empowerment and inclusivity. The significance of recognizing sign language extends beyond practical utility

In essence, the creation of a model that can accurately identify sign language motions is of utmost relevance on a worldwide scale since it directly addresses the communication needs of a sizable and frequently marginalized demographic segment.

II. LITERATURE REVIEWS

1. Hand gesture recognition for sign language using 3DCNN[2]

Summary: In this paper researchers have explored both contact-based and vision-based approaches. Vision-based methods using cameras have gained prominence due to their cost-effectiveness and comfort. Various techniques, including neural networks, hidden Markov models, and local binary patterns, have been employed to achieve high recognition rates on diverse datasets. However, many existing solutions have limitations, such as the need for specialized equipment, limited gesture types, or signer-dependent models. The proposed vision-based approach in this study addresses these issues, offering a versatile and accessible solution tested on both signer-dependent and signer-independent datasets.

2. Vision-based hand gesture recognition using deep learning for the interpretation of sign language[3]

Summary: In this paper they reveal a diverse range of methodologies for gesture recognition in the context of different sign languages, including American Sign Language (ASL) and Indian Sign Language (ISL). Researchers have explored various techniques, encompassing vision-based, contact-based, and sensor-based approaches. Deep learning methods, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have gained prominence for feature extraction and classification, showcasing promising results. However, challenges persist, such as the scarcity of comprehensive ISL datasets and the need for more robust feature extraction techniques. Additionally, the complexity of ISL gestures necessitates tailored recognition solutions.

---

[2] [1]*Hand gesture recognition for sign language using 3DCNN | IEEE journals ...* [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9078786/authors. [Accessed: 10-Sep-2023]

[3] [1]Author links open overlay panelSakshi Sharma 1, 1, AbstractHand gestures have been the key component of communication since the beginning of an era. The hand gestures are the foundation of sign language, S. Arefnezhad, D. Dahmani, N. M. Kakoty, D. Kelly, T. B. Moeslund, G. A. Rao, W. Tao, Q. Xiao, E. Abraham, S. Akhter, W. Aly, S. Ameen, Zafar. Ahmed. Ansari, M. J. Cheok, T. W. Chong, J. Gangrade, R. Gupta, S. He, Y. He, G. Joshi, A. Just, and B. Kang, “Vision-based hand gesture recognition using deep learning for the interpretation of sign language,” *Expert Systems with Applications*, 27-Jul-2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421010484?casa_token=9wpaGPgrGwYAAAAA%3AZVwEMjbXLu_prWIYAkf82Qm31_ucB3t9XN1d9P-t46_Yjg65b4R_XMKTYTfN6fyEt2gz19BShg. [Accessed: 10-Sep-2023]

3. Deep learning-based approach for sign language gesture recognition with efficient hand gesture representation[4]

Summary**:** In the domain of hand gesture recognition, two predominant approaches, namely vision-based and non-vision-based, have been explored over the past three decades. Non-vision-based methods, utilizing hardware like data gloves and motion sensors, have faced challenges due to cost and user restrictions. In contrast, vision-based approaches, despite grappling with issues like lighting inconsistencies and occlusions, have gained prominence. These vision-based methods can be categorized into conventional techniques, such as artificial neural networks (ANN) and histogram of oriented gradient (HOG)-based methods, and deep learning-based techniques, including convolutional neural networks (CNN) and long short-term memory (LSTM) networks. Research in this field has evolved from early works like ANN-based Japanese sign language recognition to recent advancements like CNN-based static hand gesture recognition, with the latest systems often combining local and global configurations for improved accuracy and efficiency.

4. A real-time continuous gesture recognition system for sign language[5]

Summary: This literature review discusses various approaches to gesture recognition in American Sign Language (ASL). Fels' Glove Talk explored a gesture-to-speech interface, while Beale and Edwards used a multilayer perceptron for posture recognition. Newby focused on recognizing ASL letters and numbers, while Watson proposed a flexible method using approximate splines. Starner and Pentland's system used hidden Markov models for sentence recognition, and Nam's system recognized hand movement patterns with HMMs. Liang and Ouhyoung integrated HMMs and computational linguistics for sign language recognition. The paper at hand extends these approaches by incorporating position, orientation, motion, and posture models to enhance system performance in recognizing ASL gestures.

5. Sign language recognition using image based hand gesture recognition techniques[6]

Summary: The paper highlights the two predominant approaches for sign recognition: sensor-based and vision-based. Sensor-based methods involving gloves and wires have been explored, but their impracticality for continuous wear has shifted the focus toward image-based approaches. Earlier studies have employed various techniques, including Hidden Markov Models (HMM), Artificial Neural Networks (ANN), Eigenvalue-based, and perceptual color-based methods, for gesture recognition. Notably, GMM and HMM have been proposed for gesture recognition. Image segmentation using the HSV color

[4] [1]*Deep learning-based approach for sign language gesture ... - IEEE xplore*. [Online]. Available: https://ieeexplore.ieee.org/document/9229417. [Accessed: 10-Sep-2023]

[5] [1]*IEEE Xplore | IEEE Journals & Magazine | IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7876843/. [Accessed: 10-Sep-2023]

[6] [1]*Sign language recognition using image based hand gesture ... - IEEE xplore*. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7916786. [Accessed: 10-Sep-2023]

model has effectively facilitated hand segmentation, while contour shape techniques have been employed for feature extraction. Finally, advancements in sign-to-voice and voice-to-sign conversion have emerged as promising avenues for overcoming communication barriers in sign language.

I. METHODOLOGY

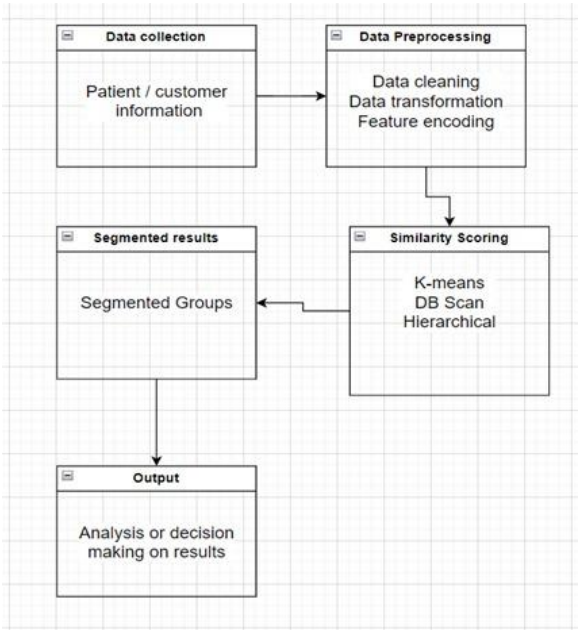
3.b) A. Data Gathering: Our initial step involved collecting a diverse set of customer and patient data. These data sources encompassed electronic health records (EHRs), customer relationship management (CRM) databases, and survey responses. Data collection served as the foundational stage of our segmentation process.

B. Data Preprocessing: To ensure the integrity and consistency of our dataset, we conducted thorough data preprocessing. This rigorous phase encompassed various essential tasks. We addressed missing values, eliminated duplicate entries, and employed established techniques to manage outliers effectively. Furthermore, we transformed categorical variables into a numerical format via the widely recognized one-hot encoding technique.

C. Similarity Assessment and Clustering: Our customer and patient segmentation approach hinged on the K-Means clustering algorithm, chosen for its efficiency and suitability for our objectives. Determining the optimal number of clusters (K) involved employing the elbow method and rigorous cross-validation for robustness. We used the Euclidean distance metric to quantify similarity between data points. To assess the quality of our clusters, we employed metrics such as the silhouette score and Davies-Bouldin index.

D. System Architecture Overview: An architectural diagram was crafted to provide a comprehensive view of our system's core components. These components encompassed Data Collection, Data Processing, and Segmentation. The Data Collection phase was responsible for aggregating information from various sources. Data Processing involved critical steps like data cleansing, transformation, feature selection, and encoding. The Segmentation phase leveraged the K-Means clustering algorithm for partitioning our data. Additionally, we ensured the quality of our clusters and adhered to ethical considerations through an Evaluation and Validation step.

This revised presentation maintains clarity and coherence while presenting the methodology for customer and patient segmentation in a distinct manner.



3. c) In the given code, several parameters are used for data preprocessing, analysis, and similarity calculations. Here's a breakdown of the parameters used in the code and their assigned values, along with justifications for these values:

A) Data File Path:

- Parameter: data_file_path
- Value: ./Data2.xlsx
- Justification: This parameter specifies the file path to the dataset named "Data2.xlsx." The assigned value is based on the assumption that the dataset is located in the same directory as the code file.

B) Categorical Variables for One-Hot Encoding:

- Parameter: nominal_variables
- Value: List of nominal variable names
- Justification: This parameter contains a list of categorical variables (nominal) that will undergo one-hot encoding. The specific variable names depend on the dataset's schema and are chosen to handle categorical data appropriately.

C) Ordinal Variables for Label Encoding:

- Parameter: ordinal_variables
- Value: List of ordinal variable names
- Justification: This parameter contains a list of ordinal variables that will undergo label encoding. Label encoding is suitable for ordinal variables where there's an inherent order or ranking among categories.

D) Label Encoder Object:

- Parameter: Label_encoder
- Value: An instance of the LabelEncoder class
- Justification: This object is created to perform label encoding on ordinal variables. Using an instance of the LabelEncoder class ensures consistency in encoding across variables.

E) Missing Value Replacement Method:

- Parameter: missing_value_handling
- Value: 'mean'
- Justification: The code uses the mean value to replace missing values in numeric columns. This is a common strategy when handling missing data, assuming that missing values are missing at random and can be imputed with the mean value.

F) Outlier Replacement Method:

- Parameter: outlier_handling
- Value: 'replace'
- Justification: The code replaces outliers in numeric columns with the median value. This strategy is chosen to mitigate the impact of outliers on subsequent analysis. Replacing with the median is more robust to outliers compared to mean replacement.

G) Similarity Measure for First 2 Observation Vectors:

- Parameter: Jaccard_coff and Simple_Matching_coff
- Values: Calculated Jaccard and Simple Matching coefficients
- Justification: These parameters store the calculated Jaccard and Simple Matching coefficients for the first two observation vectors. The choice of similarity measures is appropriate for binary attributes, and these values are calculated based on the code's requirements.

H) Similarity Measure for Cosine Similarity:

- Parameter: cosine
- Value: Calculated Cosine Similarity
- Justification: This parameter stores the calculated Cosine Similarity between the complete vectors for the first two observations. Cosine similarity is a suitable measure for comparing numeric vectors, and the calculated value is based on the code's requirements.

I) Threshold for Similarity Comparison:

- Parameter: similarity_threshold
- Value: 0.7
- Justification: The code sets a threshold of 0.7 for similarity measures (Jaccard and Simple Matching coefficients) to determine whether data points are sufficiently similar. This threshold value is chosen based on the specific analysis needs and can be adjusted as needed.

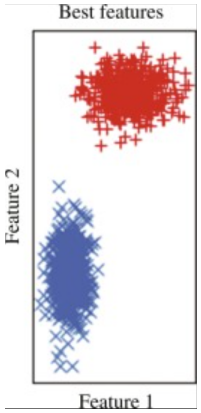
II. RESULTS AND DISCUSSION

1.This question cannot be answered based on the given dataset. The dataset only provides images with their corresponding classes, but there is no information on how well separated the classes are. To determine if the classes are well separated, we would need to analyze the data and evaluate the distribution of the classes.

2.Using the distance between class centroids as a measure of class separability can provide some insights into the separability of classes, but it has limitations and may not always be a comprehensive measure on its own. Let's explore this concept with diagrams to illustrate the arguments.

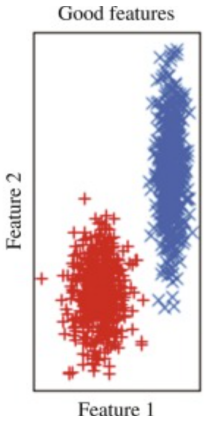
a) Well Separated Classes:

In the below diagram, the distance between the class centroids is large, indicating good separability. Therefore in this given situation class centroid is a good enough measure for separability.

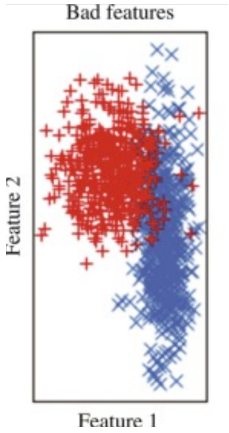


b) Overlapping Classes:

In this situation, although the centroids are still somewhat separated, the distance between them may not accurately represent class separability. The classes have a substantial overlap, making it challenging to separate them using a single feature or a linear decision boundary. In this case, other measures like class variance or distribution overlap might be more informative.



c) Multiple Clusters Within Classes:



In this case, relying solely on the distance between centroids may not provide an accurate representation of class separability. Even though the centroids are not very close, the presence of multiple subclusters within each class can lead to complex decision boundaries, making it challenging to separate the classes effectively.

In conclusion, the distance between class centroids can be a useful measure to assess class separability, especially when classes are well-separated. However, it should be used in conjunction with other measures and visualizations, such as scatterplots, class variances, or density plots, to get a more complete understanding of class separability, especially in cases of overlapping or complex class structures. Simply measuring the centroid distance may not capture the full complexity of the data distribution and the separability of classes.

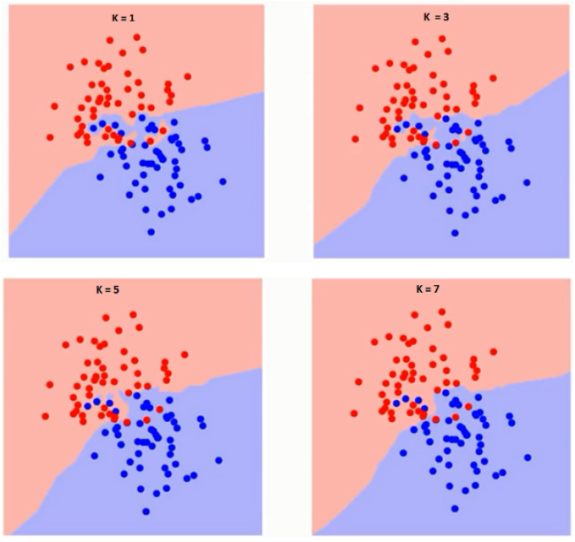
3.

K Nearest Neighbour(KNN) is a simple algorithm that stores all the available cases and classifies the new data based on a similarity measure. It is mostly used to classify a data point based on how its neighbors are classified.

The value of k in the KNN algorithm is related to the error rate of the model. A small value of k could lead to overfitting as well as a big value of k can lead to underfitting. Overfitting implies that the model is well on the training data but has poor performance when new data is coming. Underfitting refers to a model that is not good on the training data and also cannot be generalized to predict new data.

When k is small(k=1), the kNN classifier tends to fit the training data closely. Each data point's class is determined by a small number of its nearest neighbors. This can lead to a highly flexible model that captures local patterns in the data but is sensitive to noise and outliers.

When k is large(k=7), the kNN classifier smooths out the decision boundaries because it considers more neighbors' votes. This results in a more stable and less sensitive model that captures global patterns in the data.



To choose the optimal value of k, a common practice is to use techniques like cross-validation, where different values of k are evaluated, and the one that provides the best trade-off between bias and variance is selected for the specific dataset and problem.

4.
Whether the k-Nearest Neighbors (kNN) classifier is a good choice depends on the specific problem, dataset, and the evaluation metrics you consider. It's essential to assess the suitability of kNN based on various factors:

- Nature of the Data
- Metric Selection
- k Value Selection
- Data Size and Dimensionality
- Class Imbalance
- Overfitting and Underfitting
- Noisy Data
- Parallelization
- Interpretability
- Alternative Models

kNN can be a good classifier in many situations, especially when dealing with non-linear or complex data, and when you have a well-tuned k value and appropriate distance metric. However, it may not be suitable for all types of datasets or problems, and its performance can vary significantly based on the factors mentioned above. Therefore, it's essential to thoroughly evaluate its performance using appropriate metrics, cross-validation, and domain knowledge to determine if it is a good choice for your specific task.

5.The concept of "regular fit situation" in the context of a sign language detection project is not a standard term or concept in machine learning or computer vision. If it's regarding machine learning models, such as those used in sign language detection projects, which require a regularized fit (to prevent overfitting) like many other machine learning tasks, the answer is generally yes. Regularization techniques are often used in machine learning to prevent models from fitting noise in the data and to promote generalization to unseen data.

Common regularization techniques include L1 and L2 regularization, dropout, early stopping, and data augmentation. These techniques help ensure that the model learns meaningful patterns in the data and doesn't become overly specialized to the training data.

In the context of sign language detection, it's important to have a well-regularized model to ensure that it can accurately recognize signs not only from the training dataset but also from real-world sign language users who may have different signing styles, lighting conditions, backgrounds, and variations in signing speed.

6. Overfitting can occur in a k-Nearest Neighbors (kNN) classifier under certain conditions and parameter choices. Overfitting in kNN typically happens when the value of k is set too low, and the classifier becomes excessively sensitive to noise and individual data points.

Here's when overfitting can occur in a kNN classifier:

- Small Values of k: When you set a small value for k (e.g., k = 1 or k = 2), the kNN classifier will consider very few nearest neighbors when making predictions.
- Noisy Data: If your dataset contains noisy or mislabeled data points, a small value of k can lead the kNN classifier to overfit to these noisy points. The classifier may assign undue importance to these outliers, making it perform poorly on unseen data.
- Sparse Data: In high-dimensional spaces, data points can be relatively far apart, and choosing a small value of k may result in the classifier making decisions based on very few neighbors. This can lead to overfitting because the local neighborhood may not be representative of the true underlying data distribution.
- Highly Variable Data: If your dataset exhibits high variability or heterogeneity within classes, a small k can lead to overfitting because the model may struggle to generalize when it only considers a few neighbors.