www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 620-631

e-ISSN : 2583-1062
Impact Factor : 7.001

MACHINE LEARNING APPLICATION FOR PIDGIN SIGN LANGUAGE DETECTION

Biralatei Fawei[1], Ide Mercy Azibaye[2], Patrick Kenekayoro[3], Bunikiye Richard Japheth[4]
[1,2,3,4]Department Of Computer Science, Niger Delta University, Wilberforce Island, Bayelsa State, Nigeria.

ABSTRACT

Sign language is a form of non-verbal communication that is used by people having difficulty in speech or hearing. There is always a need to translate between sign language and text for communication between the hearing impaired and those without hearing difficulties. Nigerian Schools adopt the American Sign Language, but locally the Pidgin Sign Language is more predominant. Although there have been studies that have used machine learning methods to translate sign language to text, there is a dearth of research that has focused on the local (Nigerian) Pidgin Sign Language (PSL). Thus, it is unknown how well the state-of-the-art methods for sign language detection and translation will perform for the Nigerian Pidgin Sign Language. This research aims to bridge this gap by investigating how well these state-of-the-art techniques tailored for American Sign Language (ASL) will perform for Nigerian Pidgin Sign Language (PSL) and adapt these techniques for better performance. Experimental and engineering research design was adopted in this research. The multilayer Perceptron (MLP) methodology was implemented and it is underpinned by the use of lightweight neural classifiers and temporal smoothing techniques, ensuring the system's performance is both robust and suitable for real-time operation on standard, non-specialized hardware. A webcam was used for data capturing for a total of 1,000 video frame gestures in PSL for training and testing for 8 different gestures summing to 8,000 frames using Python and OpenCV. The Static Classifier's superior performance accuracy of 94.15% confirms that the method of extracting and normalizing static keypoint data is highly effective for recognizing static PSL signs while the significant underperformance of the Temporal Classifier with an accuracy of 24.39% points to a critical issue in the dynamic gesture recognition pipeline. However, the model recognizes the various hand motions and outputs the appropriate word on the screen.

Keywords: Machine learning, Pidgin sign Language, Long-short Term Memory, Recurrent Neural Network, Gesture recognition.

1. INTRODUCTION

Communication is an act of sharing or exchanging information, feelings, or ideas that is essential for people, this communication is achieved when both parties understand a common language, in the case of deaf and dumb, it is achieved through sign language (Patil, 2021). Sign Language or hand speak is a language that uses hand movement to show or express words and alphabets. However, the use of any other finger spelling movements representing other alphabets can be easily adopted. Elmahgiubi et al. (2015) stated that Sign language has several components that can be studied collectively or individually in combination with computer engineering, particularly within the context of facial recognition and body movements. This is because computers and facial recognition systems play a major role in supporting the deaf to communicate with intelligent machines.

Sign language is categorized into village and deaf sign language (Meir et al. 2010). The author described the village sign as a sign language developed within small communities or villages where transmission is within and between families. For example, Martin's Vineyard sign language in the United Kingdom, Ban Khor sign language in Thailand, Adamarobe sign language in Ghana etc. While the deaf community sign language came to bear by bringing together unrelated signers from different backgrounds in locations such as cities or schools. This research adopts the American Sign Language (ASL), due to its popularity and major use in Nigeria and globally. Also, word representation in ASL is usually done by loanwords from English and it represents phonemics using expressions and movement in the face, the upper body, and the hands. Pidgin sign language (PSE), also called pidgin sign English may be seen as an initial creole. PSE can be learnt at almost any age by a deaf person. Generally, it is worth noting that there is no good definition for pidgin and creole, but it is agreed that pidgin languages are reduced in structure and also contain a partial mixture of two to several languages that contain structure common to none of the languages in the communication system. One major variation in PSE is the use of articles, plurality, progressive aspect (verb reduplication), and copula. PSE is mostly used by older people. Generally, it has an uninflected copula. It is interesting to note that the presence of the past tense copula forms in Pidgin Sign English implies the presence of the present tense form. For example, be + ing. PSE however, drops the redundant ing. In summary, most Pidgin language articles are deleted. The copula is usually inflected such as lost English plural and most derivations are lost just as they are in PSE (Woodward, 1973).

Pidgin Sign Language (PSL) is a vibrant tapestry of gestures and signs that serve as a vital communication tool for deaf and dumb as well as hard-of-hearing communities across West Africa and beyond. However, a persistent challenge remains - the limited accessibility of PSE for those who do not understand it. This creates a significant barrier to communication and social interaction. Hence, this study applies machine learning technique for pidgin sign language detection. Machine learning (ML) emerges as a beacon of hope, offering the potential to bridge this gap through sign language detection applications (Zheng et al., 2020). By harnessing the power of ML algorithms, we can create systems that can interpret PSE signs, fostering seamless communication between deaf and hearing individuals.

Machine learning which is a sub-class of artificial intelligence is one of the current areas in the field of Computer Science. Arthur Samuel who was one of the pioneers of Artificial intelligence (AI) research, introduced the term machine learning in computer gaming (Sousa Antunes et al., 2024). Machine learning (ML) practices with a set of statistical algorithms from a large database containing analyzed and interpreted data made available as input, enables the system to learn from it, and based on the learned information make the best probable decision. Some important areas of ML application in real-world include image interpretation, human behavioral patterns and identity, text and speech recognition, crime and fraud detection, bank and financial sector, biomedical and medical analysis and so on (El Naqa & Murphy, 2015). Deep neural network which is a subcategory of machine learning, is a network model with neurons having several parameters and layers in between input and output. Its complete architecture is used for feature extraction and alteration processes. Application areas of deep learning include language understanding and biometrics, computer vision, speech recognition, personalization for solutions in particular cases, etc. (Sharma et al., 2021).

2. RELATED WORKS

Kapuscinski & Wysocki, (2001) created a gesture recognition system that can identify numbers 0 through 9. They used the Hit-miss transform to predict hand postures and performed skin tone-based hand segmentation, utilizing normalized color space. With a training set of 600 gestures these hand postures were utilized to train two, three, and four-state HMMs. The training accuracy was 98% on the training set and 97.2% on the test set. A gesture identification system using Gabor features and fuzzy-c-mean clustering was proposed in (Amin & Yan, 2007). By manually reducing the hand region and using a dataset of hand photos with a constant light background, they were able to eliminate hand segmentation. For these photos, Gabor filters were used along-side with the principal component analysis (PCA) to reduce the dimensionality as a part of feature extraction. The extracted features are then used to train the fuzzy-c-mean clustering algorithm. In an experimental based cross-validation test, this system's accuracy score was 93.2%. Garg et al., (2009) adopted a similar strategy to create a dataset of 1320 photos utilizing 11 signers and a set of 11 hand motions in their research. They used normalized RGB color and skin-based segmentation for hand segmentation. When put to test with 660 sets of photos, the system produced an accuracy of 95.2%. Lee et al. (2016) takes a similar approach but used a jump motion sensor instead of a network glove or Kinect sensor. The data collected by the jumping motion sensor was used to train both the KNearest and SVM neighborhood classifiers with approximately 7,900 images. This technique achieves an accuracy of 72.7 and 79.7% for KNN and SVM respectively. Admasu & Raimond, (2010) developed the Ethiopian Sign Language (ESL) recognition system, using a neural network produces a recognition rate of 98.5%. In this research, the authors used manually pre-processed hand images and performed basic hand segmentation to obtain the hand region. Gabor filter is applied to extract features and form a three-layer neural network. The network architecture consists of 20 input neurons, 100 hidden layer neurons, and 34 output neurons. Hasan et al., (2022) used neural network-based ASL recognition using background subtraction based on a Gaussian mean model for hand segmentation. Shape descriptors and motion vectors were extracted from the segmented frames and used to train the three-layer neural network. The authors claim to achieve a 100% recognition rate for A-to-Z gesture recognition. Patil, (2021) designed a real-time sign language detector for Indian Sign Language using Convolutional Neural Network (CNN), Support Vector Machine (SVM) and K-nearest neighbor classifier (KNN). The author used a Tensor Flow object detection API while image capture was done using the webcam. The models with small data set achieved a good degree of accuracy. Angona et al., (2020) discussed how an interpreter can help translate Bangladeshi Sign Language into a computer-understandable format. The author proposed a Bangladeshi sign language recognition system based on computer vision (BDSL) where PCA (Principal Component Analysis) was used and tested for 6 Bengali vowels and 10 numbers in Bengali. Wang et al. (2022) in their study proposed a smart glove that integrates flexible sensors and an inertial measurement unit (IMU) used for gesture recognition. The prototype was built according to the hand movement in three dimensions. The app has also been used in gaming, robotics and in the medical field. Though, it was initially tested to convert Indian Sign Language into speech. A new sign language learning system based on sampling and stitching 2D images to solve the problem of conventional symbol recognition was implemented in Núñez-Marcos et al., (2023). The system used a sample data from a sign language demonstration video learned by a custom convolutional neural network. The classification was performed by a KNN algorithm using Euclidean distance measure. The results obtained in the experiment have high accuracy with a data set of only 20 cases. Asonye et al. (2018) observed that over four years the Nigerian Deaf Community (NDC) has more than 75% of young deaf youth from birth to adulthood in Nigeria with virtually no access to marked language. Citing that more than 95% of them are provided by auditory tutors without marked language information and without an early mediation program to work with but rather remained with the early acquisition of skills by mother tongue. This group of deaf children have language difficulties and language gaps leading to very low thinking and semantic abilities, and no academic language that lays the foundation for progress at home and at school. Implementation of an early detection and early reconciliation programs are the missing answers to the problem of tongue holes in deaf children in Nigeria. A study was carried out in the Bura-speaking region of Upper Eastern Nigeria and surveyed an unreported deaf language in the town of Kukurpu (Blench et al., 2006). Asonye et al. (2020) states that the Nigerian Sign Language Project has been shown to attempt to support the development of Nigerian Signature Communication (NSL) of symbols and movements used by deaf Nigerians themselves. In addition to providing new pedagogical approaches at school, preparation and support will also be provided to families with youth with hearing loss. However, in all of these researches there is no singular one tailored to solving pidgin sign language gesture recognition.

3. METHODOLOGY

This section details the systematic methodological framework for the development and evaluation of the Pidgin Sign Language Recognition System (PSLRS). The PSLRS is a real-time computer vision application designed to identify and classify eight isolated hand gestures from Pidgin Sign Language (PSL). Given the absence of a standardized, large-scale dataset for PSL, this research adopts a rigorous approach that spans the entire machine learning pipeline: from custom data acquisition and preprocessing to a novel model design and comprehensive evaluation. The methodology is underpinned by the use of lightweight neural classifiers and temporal smoothing techniques, ensuring the system’s performance is both robust and suitable for real-time operation on standard, non-specialized hardware. This structured approach not only addresses the specific challenges of PSL (a simplified subset of Nigerian Sign Language (NSL) with Pidgin English-inspired vocabulary) but also ensures the scientific rigor and practical applicability of the research.

3.1 Research Design

The study employs a combined approach of experimental and engineering research design, focusing on both the practical development of a novel machine learning system and the quantitative evaluation of its performance. This approach is particularly suited to a computer vision task where the primary goal is the creation of a functional, real-time application. The methodology relies on a quantitative paradigm, utilizing normalized hand landmark coordinates to train and test two distinct Multilayer Perceptron (MLP) classifiers. The first classifier handles static handshapes, while the second processes dynamic gestures, with a 16-frame smoothing window to enhance accuracy. This dual-classifier design is uniquely tailored to the mixed nature of PSL signs, which include both stationary and motion-based gestures. The design also strategically addresses common computer vision challenges such as variability in lighting conditions and diverse hand orientations by using normalization and a robust feature extraction method.

3.2 Data Acquisition

The development of the Pidgin Sign Language Recognition System (PSLRS) necessitated the creation of a custom dataset, as no publicly available PSL datasets existed at the time of this research. For the purpose of this task eight proficient PSL signers were recruited from Alderstown School of the Deaf, Delta State and Bayelsa State School for children with special needs Opolo Yenagoa all in Nigeria. The participants were verified by experts in the selected schools. The participant group was diversified by age, gender, and hand characteristics to maximize the model's ability to generalize to a wider population. All participants provided informed consent, and ethical approval for the study. This dataset creation process was carefully planned and executed under controlled conditions to ensure high data quality, consistency, and reproducibility. The controlled environment minimized variability in lighthening, background, and camera placement, making the dataset more reliable for training and evaluating the recognition models. Video was captured at 960x540 pixels and 30 frames per second (FPS) using a standard webcam. Processing was performed on a system equipped with an Intel Core i5 CPU and an NVIDIA GPU for accelerated inference. In general, a standardized procedure was followed for capturing each gesture, ensuring that every sample adhered to uniform guidelines. The dataset was carefully designed to recognize eight specific Pidgin Sign Language (PSL) gestures: "come", "chop", "my", "ball", "stop", "there", "here", and "bye bye." These gestures were chosen because of their high frequency and communicative value in everyday Pidgin English interactions, reflecting common expressions in Nigerian Sign Language communities. To ensure both linguistic and cultural accuracy, the selected gestures were reviewed and validated by expert signers in the selected institution. Participants performed each of the eight signs multiple times while the PSLRS application operated in a logging mode. The MediaPipe Hands library was used to extract 21 keypoints per hand. These keypoints were then logged into two separate CSV files: keypoint.csv for static handshapes and point_history.csv for dynamic fingertip trajectories. A total of 8,000 samples were collected, with about 1,000 samples per sign. The dynamic data was captured by specifically tracking the index finger's landmark (landmark 8) across a 16-frame temporal window as shown in Figure 3.1. The structure of the data file is visualized in Figure 3.2 and their content in Figure 3.3 respectively. Each collected sample was manually labelled with a sign ID (0-7) at the time of collection. Static gesture samples were stored as 42-dimensional vectors ($f \in R^{42}$), derived from the x and y coordinates of 21 keypoints. Dynamic gesture samples were stored as 32-dimensional vectors ($h \in R^{32}$), representing the x and y coordinates of the index fingertip over 16 frames. No raw video files were stored, prioritizing a lightweight data format suitable for efficient processing.

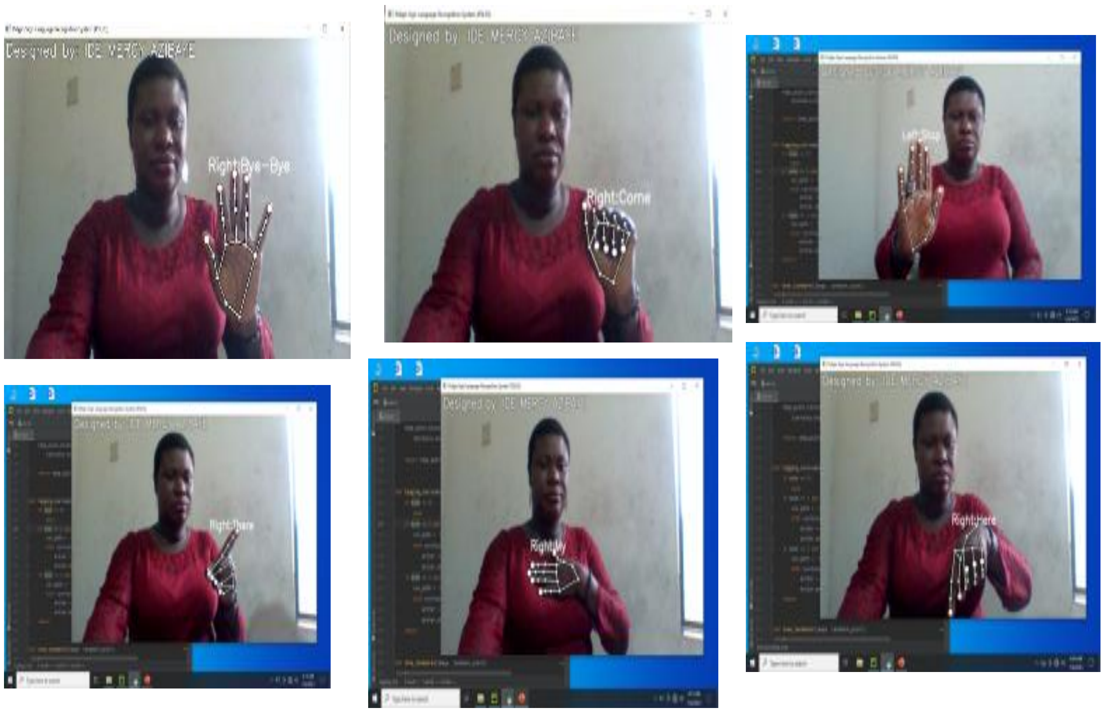
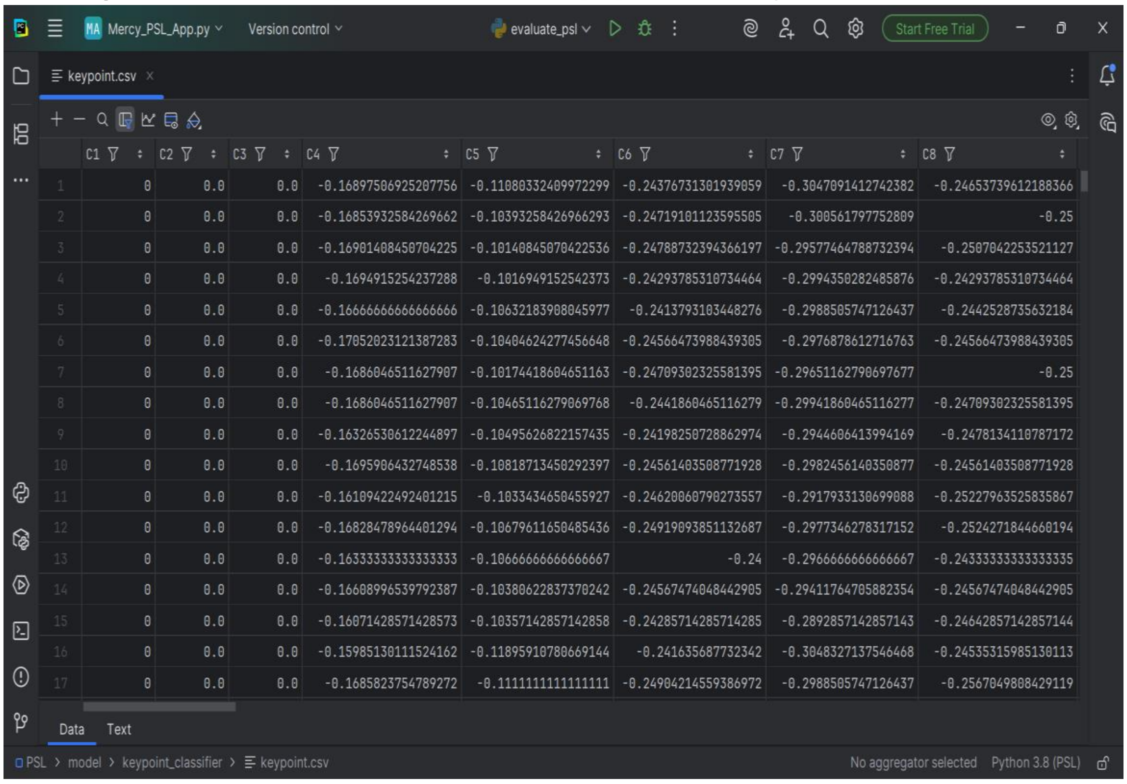
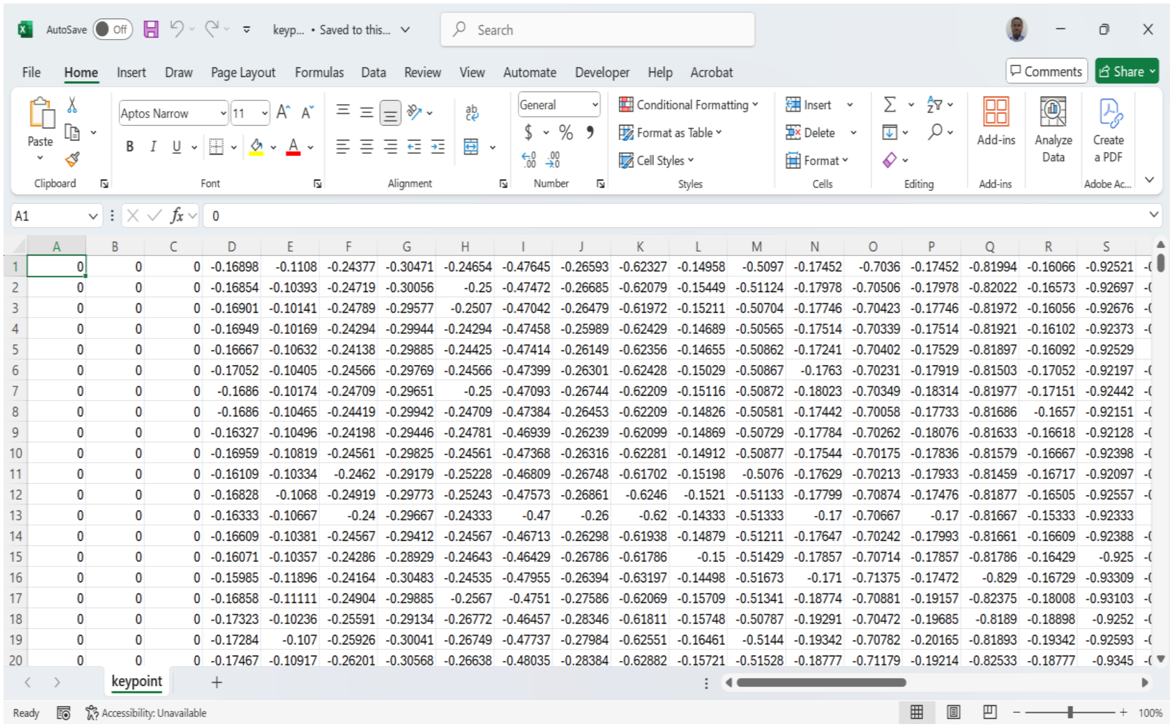


Figure 3.1: Different Input of Gestures for the purpose of training the model



	C1	C2	C3	C4	C5	C6	C7	C8
1	0	0.0	0.0	-0.16897506925207756	-0.11080332409972299	-0.24376731301939059	-0.3047091412742382	-0.24653739612188366
2	0	0.0	0.0	-0.16853932584269662	-0.10393258426966293	-0.24719101123595505	-0.300561797752809	-0.25
3	0	0.0	0.0	-0.16901408450704225	-0.10140845070422536	-0.24788732394366197	-0.29577464788732394	-0.2507042253521127
4	0	0.0	0.0	-0.1694915254237288	-0.1016949152542373	-0.24293785310734464	-0.2994350282485876	-0.24293785310734464
5	0	0.0	0.0	-0.16666666666666666	-0.10632183908045977	-0.2413793103448276	-0.2988505747126437	-0.2442528735632184
6	0	0.0	0.0	-0.17052023121387283	-0.10404624277456648	-0.24566473988439305	-0.2976878612716763	-0.24566473988439305
7	0	0.0	0.0	-0.1686046511627907	-0.10174418604651163	-0.24709302325581395	-0.29651162790697677	-0.25
8	0	0.0	0.0	-0.1686046511627907	-0.10465116279069768	-0.2441860465116279	-0.29941860465116277	-0.24709302325581395
9	0	0.0	0.0	-0.16326530612244897	-0.10495626822157435	-0.24198250728862974	-0.2944606413994169	-0.2478134110787172
10	0	0.0	0.0	-0.1695906432748538	-0.10818713450292397	-0.24561403508771928	-0.2982456140350877	-0.24561403508771928
11	0	0.0	0.0	-0.16109422492401215	-0.1033434650455927	-0.24620060790273557	-0.2917933130699088	-0.25227963525835867
12	0	0.0	0.0	-0.16828478964401294	-0.10679611650485436	-0.24919093851132687	-0.2977346278317152	-0.2524271844660194
13	0	0.0	0.0	-0.16333333333333333	-0.10666666666666667	-0.24	-0.29666666666666667	-0.24333333333333335
14	0	0.0	0.0	-0.16608996539792387	-0.10380622837370242	-0.24567474048442905	-0.29411764705882354	-0.24567474048442905
15	0	0.0	0.0	-0.16071428571428573	-0.10357142857142858	-0.24285714285714285	-0.2892857142857143	-0.24642857142857144
16	0	0.0	0.0	-0.15985130111524162	-0.11895910780669144	-0.241635687732342	-0.3048327137546468	-0.24535315985130113
17	0	0.0	0.0	-0.1685823754789272	-0.1111111111111111	-0.24904214559386972	-0.2988505747126437	-0.2567049808429119

Figure 3.2: View of the custom Pidgin Sign Language (PSL) dataset structure as displayed in the PyCharm IDE



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	0	0	0	-0.16898	-0.1108	-0.24377	-0.30471	-0.24654	-0.47645	-0.26593	-0.62327	-0.14958	-0.5097	-0.17452	-0.7036	-0.17452	-0.81994	-0.16066	-0.92521
2	0	0	0	-0.16854	-0.10393	-0.24719	-0.30056	-0.25	-0.47472	-0.26685	-0.62079	-0.15449	-0.51124	-0.17978	-0.70506	-0.17978	-0.82022	-0.16573	-0.92697
3	0	0	0	-0.16901	-0.10141	-0.24789	-0.29577	-0.2507	-0.47042	-0.26479	-0.61972	-0.15211	-0.50704	-0.17746	-0.70423	-0.17746	-0.81972	-0.16056	-0.92676
4	0	0	0	-0.16949	-0.10169	-0.24294	-0.29944	-0.24294	-0.47458	-0.25989	-0.62429	-0.14689	-0.50565	-0.17514	-0.70339	-0.17514	-0.81921	-0.16102	-0.92373
5	0	0	0	-0.16667	-0.10632	-0.24138	-0.29885	-0.24425	-0.47414	-0.26149	-0.62356	-0.14655	-0.50862	-0.17241	-0.70402	-0.17529	-0.81897	-0.16092	-0.92529
6	0	0	0	-0.17052	-0.10405	-0.24566	-0.29769	-0.24566	-0.47399	-0.26301	-0.62428	-0.15029	-0.50867	-0.1763	-0.70231	-0.17919	-0.81503	-0.17052	-0.92197
7	0	0	0	-0.1686	-0.10174	-0.24709	-0.29651	-0.25	-0.47093	-0.26744	-0.62209	-0.15116	-0.50872	-0.18023	-0.70349	-0.18314	-0.81977	-0.17151	-0.92442
8	0	0	0	-0.1686	-0.10465	-0.24419	-0.29942	-0.24709	-0.47384	-0.26453	-0.62209	-0.14826	-0.50581	-0.17442	-0.70058	-0.17733	-0.81686	-0.1657	-0.92151
9	0	0	0	-0.16327	-0.10496	-0.24198	-0.29446	-0.24781	-0.46939	-0.26239	-0.62099	-0.14869	-0.50729	-0.17784	-0.70262	-0.18076	-0.81633	-0.16618	-0.92128
10	0	0	0	-0.16959	-0.10819	-0.24561	-0.29825	-0.24561	-0.47368	-0.26316	-0.62281	-0.14912	-0.50877	-0.17544	-0.70175	-0.17836	-0.81579	-0.16667	-0.92398
11	0	0	0	-0.16109	-0.10334	-0.2462	-0.29179	-0.25228	-0.46809	-0.26748	-0.61702	-0.15198	-0.5076	-0.17629	-0.70213	-0.17933	-0.81459	-0.16717	-0.92097
12	0	0	0	-0.16828	-0.1068	-0.24919	-0.29773	-0.25243	-0.47573	-0.26861	-0.6246	-0.1521	-0.51133	-0.17799	-0.70874	-0.17476	-0.81877	-0.16505	-0.92557
13	0	0	0	-0.16333	-0.10667	-0.24	-0.29667	-0.24333	-0.47	-0.26	-0.62	-0.14333	-0.51333	-0.17	-0.70667	-0.17	-0.81667	-0.15333	-0.92333
14	0	0	0	-0.16609	-0.10381	-0.24567	-0.29412	-0.24567	-0.46713	-0.26298	-0.61938	-0.14879	-0.51211	-0.17647	-0.70242	-0.17993	-0.81661	-0.16609	-0.92388
15	0	0	0	-0.16071	-0.10357	-0.24286	-0.28929	-0.24643	-0.46429	-0.26786	-0.61786	-0.15	-0.51429	-0.17857	-0.70714	-0.17857	-0.81786	-0.16429	-0.925
16	0	0	0	-0.15985	-0.11896	-0.24164	-0.30483	-0.24535	-0.47955	-0.26394	-0.63197	-0.14498	-0.51673	-0.171	-0.71375	-0.17472	-0.829	-0.16729	-0.93309
17	0	0	0	-0.16858	-0.11111	-0.24904	-0.29885	-0.2567	-0.4751	-0.27586	-0.62069	-0.15709	-0.51341	-0.18774	-0.70881	-0.19157	-0.82375	-0.18008	-0.93103
18	0	0	0	-0.17323	-0.10236	-0.25591	-0.29134	-0.26772	-0.46457	-0.28346	-0.61811	-0.15748	-0.50787	-0.19291	-0.70472	-0.19685	-0.8189	-0.18898	-0.9252
19	0	0	0	-0.17284	-0.107	-0.25926	-0.30041	-0.26749	-0.47737	-0.27984	-0.62551	-0.16461	-0.5144	-0.19342	-0.70782	-0.20165	-0.81893	-0.19342	-0.92593
20	0	0	0	-0.17467	-0.10917	-0.26201	-0.30568	-0.26638	-0.48035	-0.28384	-0.62882	-0.15721	-0.51528	-0.18777	-0.71179	-0.19214	-0.82533	-0.18777	-0.9345

Figure 3.3: Excel view of the PSL dataset, presenting extracted hand landmark coordinates and temporal movement sequences used for gesture recognition.

3.3 Data Preprocessing

Raw video frames were systematically processed to extract meaningful features for classification (see Figure 3.4). The process involves frame-by-frame hand landmark detection, normalization of landmark coordinates to account for variations in hand size and position, as well as temporal feature encoding using a sliding 16-frame window. The preprocessing pipeline converts raw visual data into compact numerical representations optimized for gesture recognition models.

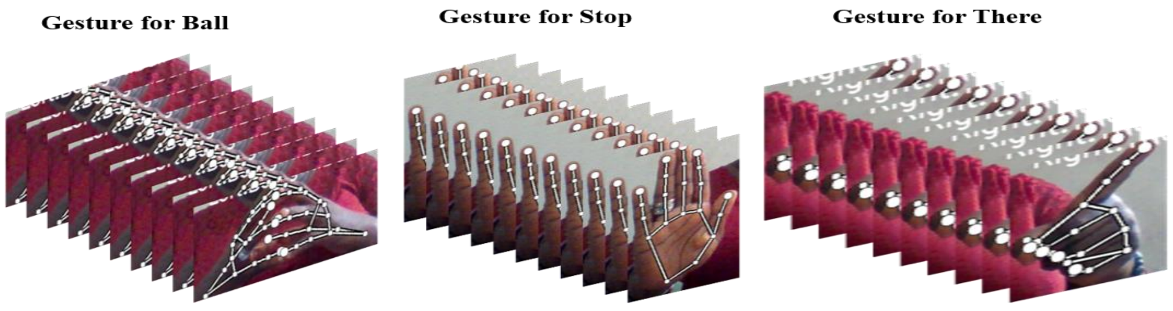


Figure 3.4: Visualization of ten sampled frames per sign for three representative signs: "ball," "stop," and "there."

Each column corresponds to a unique sign, while the rows display ten sequential frames illustrating the temporal hand movements that characterize video frames were extracted using OpenCV's VideoCapture functionality, with each frame flipped horizontally to create a mirrored view consistent with natural human perception during interaction. Hand landmarks were detected using MediaPipe Hands, which identifies up to two hands per frame and outputs 21 keypoints per detected hand. Each landmark is represented as:

$$L=[(x_i,y_i)]_{i=0}^{20} \tag{1}$$

The dataset L contains 21 data points because it starts at i=0, and ends at i=20

Detection and tracking confidence thresholds were set to 0.7 and 0.5, respectively (Lugaresi et al., 2019). To ensure positional invariance and scale consistency, a three-step normalization process was applied:

1. **Translation relative to wrist**: For each landmark i (where i=0,1,…,20), the wrist coordinates (x_0,y_0) were subtracted from each landmark coordinate (x_i,y_i):

$$x_i'=x_i-x_0,\ \ y_i'=y_i-y_0 \tag{2}$$

2. **Scale by maximum absolute value**: The maximum absolute value m across all translated coordinates were computed with:

$$m=\max\left(\{|x_i'|,|y_i'|\}\right) \qquad (3)$$

Each coordinate was then normalized by dividing it by m:

$$\hat{x}_i=\frac{x_i'}{m},\ \hat{y}_i=\frac{y_i'}{m} \qquad (4)$$

3. **Flatten to feature vector**: The normalized coordinates for all the 21 landmarks were concatenated into a single feature vector as:

$$f\in R^{42} \qquad (5)$$

These vectors captured the relative positions of the 21 keypoints as a flattened representation suitable for static sign classification. Dynamic gestures were encoded by tracking the position of the index finger tip (landmark 8) across a temporal window of 16 frames. The coordinates for each frame were stored in a deque of fixed length with placeholder coordinates [0,0], inserted for frames where the gesture was classified as static (i.e., sign ID $\neq 2$) to maintain sequence consistency. Each point in the sequence was normalized relative to the base point (coordinates of the index finger in the first frame, (x_b,y_b) and the image dimensions (W,H):

$$x'=\frac{x-x_b}{W},\ y'=\frac{y-y_b}{H} \qquad (6)$$

The normalized coordinates were concatenated into a single feature vector:

$$h\in R^{32} \qquad (7)$$

These vectors captured the trajectory of the index finger over 16 frames and provide temporal information critical for dynamic sign recognition. The acquired dataset comprises of 8000 samples (1000 per sign) which was divided into training, validation, and test sets with a split of 70:15:15 ratio. This corresponds to 5600 samples for training, 1200 for validation, and 1200 for testing, ensuring balanced evaluation and preventing overfitting.

3.4 Windowed MLP Architecture

The research employed a dual-classifier architecture built on two complementary Multilayer Perceptron (MLP) models for the Pidgin Sign Language Recognition System (PSLRS). This design was adopted to balance recognition accuracy with low-latency performance, making the system suitable for real-time deployment on commodity hardware (see overall structure of the model in Figure 3.5). The KeyPointClassifier is responsible for recognizing static handshapes. Its input is a 42-dimensional normalized landmark vector ($f\in R^{42}$). It captures the precise spatial configuration of 21 hand landmarks at a single point while the PointHistoryClassifier is specialized in recognizing dynamic gestures using a 32-dimensional normalized point history vector ($h\in R^{32}$). It encodes the fingertip trajectory of the index finger over a 16-frame temporal window. Together, these classifiers enable robust recognition of both static and motion-based signs. Both models are pre-trained and implemented in TensorFlow Lite, enabling efficient inference on devices with limited computational resources. Their architecture consists of an input layer sized to match the dimensionality of their respective feature vectors (42 neurons for the KeyPointClassifier and 32 neurons for the PointHistoryClassifier).

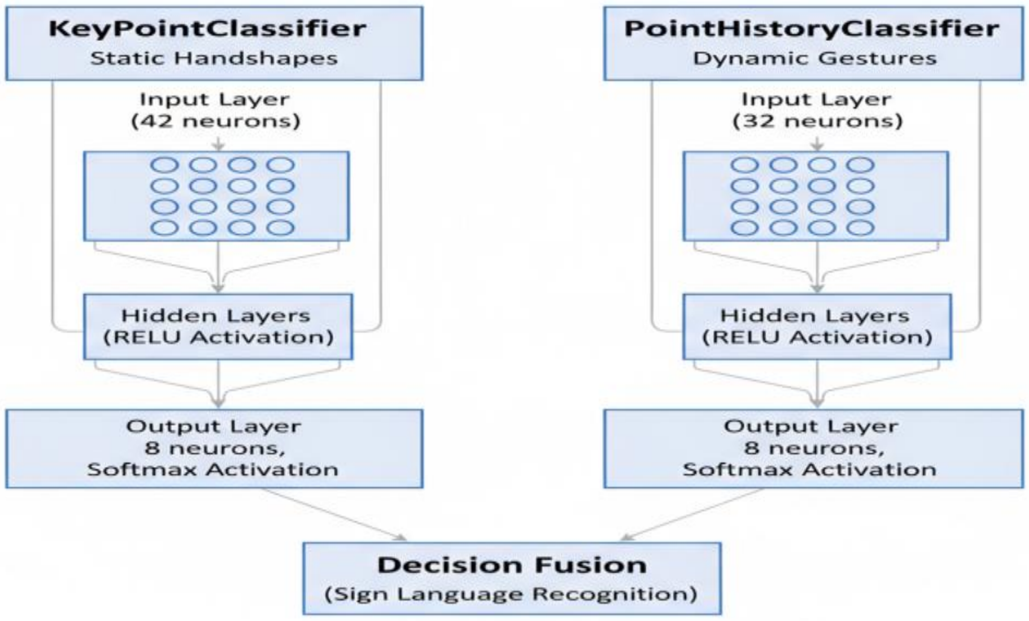

KeyPointClassifier
Static Handshapes
Input Layer (42 neurons)
Hidden Layers (RELU Activation)
Output Layer 8 neurons, Softmax Activation
PointHistoryClassifier
Dynamic Gestures
Input Layer (32 neurons)
Hidden Layers (RELU Activation)
Output Layer 8 neurons, Softmax Activation
Decision Fusion
(Sign Language Recognition)

Figure 3.5: Block diagram of the PSLRS showing two MLP classifiers for static and dynamic gesture recognition with temporal smoothing.

They are followed by one or two fully connected hidden layers with Rectified Linear Unit (ReLU) activation function and a neuron count ranging from 64 to 128 neurons. The final layer in both classifiers is a Softmax layer with eight output neurons, corresponding to the eight target Pidgin Sign Language (PSL) signs. For dynamic gesture recognition, the system incorporates a post-processing step known as temporal smoothing to improve classification stability. This is achieved by applying majority voting over a sequence of 16 predictions, defined mathematically as:

$$g=\mathrm{mode}\left(g_{hist}\right) \qquad (8)$$

where g_{hist} represents the prediction history across the 16-frame window. Python's Counter.most_common () function is used to identify the most frequent class within this sequence, reducing the impact of noisy frame-level predictions. By combining efficient architecture design with temporal smoothing, the PSLRS achieves both responsiveness and accuracy in real-time recognition tasks. The system's real-time functionality is built by integrating several open-source libraries (see Table 3.1).

Table 3.1: Summary of tools, libraries, and their functions in the development of the PSLRS.

Library/Tool	Functionality
OpenCV	Used for core functionalities such as video capture, frame processing, and on-screen visualization.
MediaPipe Hands	The primary library for performing real-time hand landmark detection.
TensorFlow Lite	The chosen framework for implementing the lightweight MLP classifiers, ensuring efficient on-device inference.
NumPy	Leveraged for various numerical computations required during data preprocessing and feature extraction.
Tkinter/PIL	Used to develop the graphical user interface (GUI) for displaying instructions and sign images to the user.
CSV	Utilized for data logging and the storage of training data.

3.5 Model Training and Optimization

Model training is conducted offline using the prepared CSV datasets, which provide the labelled training data for the MLPs. The model is trained using supervised learning with a categorical cross-entropy loss function defined as:

$$\mathcal{L} = -\sum_{i} y_i \log(\hat{y}_i) \qquad (9)$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for each class. The Adam optimizer was employed with a learning rate of approximately 0.001. Training runs for 50–100 epochs with a batch size of 32, suitable for lightweight models to balance convergence speed and computational efficiency. While early stopping based on validation loss prevents overfitting and normalization reduces variance. In this research, hyperparameter tuning was manually done via validation performance. However, the graphics processing unit (NVIDIA GPU) accelerates the training and inference ensuring low latency. The overall system design is illustrated in the Process Block Diagram (Figure 3.6), which highlights the modular integration of computer vision, machine learning, and user interface components.

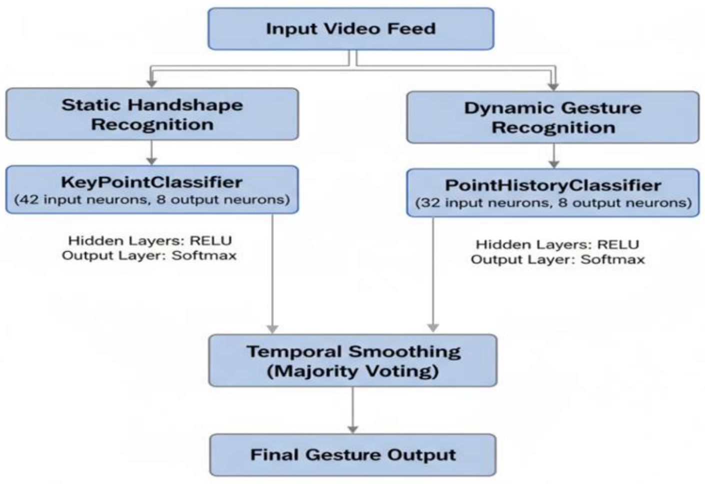

Input Video Feed
Static Handshape Recognition
Dynamic Gesture Recognition
KeyPointClassifier (42 input neurons, 8 output neurons)
PointHistoryClassifier (32 input neurons, 8 output neurons)
Hidden Layers: RELU
Output Layer: Softmax
Hidden Layers: RELU
Output Layer: Softmax
Temporal Smoothing (Majority Voting)
Final Gesture Output

Figure 3.6: Block Process Diagram for the Pidgin Sign Language Recognition System

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 620-631
www.ijprems.com
editor@ijprems.com
e-ISSN : 2583-1062
Impact Factor : 7.001

Predicted signs are visualized in real time through OpenCV overlays, which display both tracked landmarks and recognized sign labels. A Tkinter-based graphical user interface (GUI) further enhances this experience by providing sign references, project details, and step-by-step instructions in Pidgin English. The complete implementation of this pipeline includes the data flow and component integration which is provided in the Python source code file Merscy_PSL_App.py. The GUI is intentionally simple and accessible, featuring illustrative sign images, such as pictures/come.png, and clear usage instructions. It also offers straightforward controls, allowing users to exit the recognition session instantly by pressing the ESC key. The PSLRS achieves a runtime latency of under 100 milliseconds per frame. This performance ensures smooth, natural interaction and demonstrates the system's ability to deliver reliable recognition on consumer-grade hardware.

4. SYSTEM EVALUATION

This section presents a comprehensive evaluation of the Pidgin Sign Language Recognition System (PSLRS) developed to detect and classify hand gestures in real-time for pidgin sign language communication. The evaluation focuses on the performance of two classifiers: the Static Classifier (KeyPointClassifier), which handles static hand signs, and the Temporal Classifier (PointHistoryClassifier), which recognizes dynamic gestures. A key component of the evaluation is an ablation study that quantifies the impact of temporal smoothing on dynamic gesture recognition. All performance metrics—including accuracy, precision, recall, and F1-score—are benchmarked against expected values, and the system's real-time capability is measured in frames per second (FPS). The results presented are derived from data collected by the PSLRS application itself and processed offline using a dedicated evaluation script. The Static Classifier was evaluated on the keypoint.csv dataset. The Temporal Classifier was evaluated on the point_history.csv dataset, with a dedicated ablation study comparing its performance both with and without the temporal smoothing mechanism (majority voting over 16 frames). Finally, the system's real-time performance was measured by capturing and processing frames for a 10-second duration using the same camera and processing pipeline as the main application. This provided an average FPS, which serves as a crucial metric for a real-time system. All evaluation results and visualizations were programmatically generated and saved to the evaluation_results directory.

5. RESULTS AND DISCUSSION

The evaluation results obtained from the evaluate_psl.py script are summarized in table 5.1 and 5.2. They presents a detailed analysis of the performance of both classifiers, the ablation study, and the system's FPS. The Static Classifier, which relies on normalized keypoint data to classify static hand signs, demonstrated exceptional performance surpassing all predefined benchmarks gleaned in Table 5.1.

Table 5.1: Evaluation Metrics for the KeyPointClassifier

Metric	Result	Expected Benchmark
Accuracy	0.9415	~0.90
Precision	0.9361	~0.89
Recall	0.9375	~0.90
F1-Score	0.9355	~0.89

With an accuracy of **94.15%**, the model proved highly effective at distinguishing between static hand gestures. The consistently high scores for precision, recall, and F1-score (all above 93%) indicates robust and balanced performance across all classes of static signs.

The Temporal Classifier, designed to recognize dynamic gestures using point history data exhibited performance significantly below the expected benchmarks as seen in Table 5.2.

Table 5.2: Evaluation Metrics for the Point History Classifier

Metric	Result	Expected Benchmark
Accuracy	0.2439	~0.93
Precision	0.1895	~0.92
Recall	0.1685	~0.93
F1-Score	0.1375	~0.92

The low accuracy of **24.39%** and corresponding low scores for other metrics suggest that the model struggled to correctly classify dynamic gestures. This result indicates a fundamental issue which may be related to data quality, class imbalance, or a mismatch in the preprocessing pipeline.

An ablation study was conducted to determine the effect of the temporal smoothing mechanism on the Temporal Classifier's performance.

- Accuracy without voting: **0.8132**
- Accuracy with voting: **0.2439**
- Performance change: **-56.93%** (Expected: +5–10%)

Contrary to the hypothesis that temporal smoothing would improve accuracy, it resulted in a significant 56.93% decrease in performance. This surprising outcome suggests that the majority voting mechanism in its current form is a source of error rather than an enhancement, which is likely due to inconsistencies in the collected data or a voting window that is too long for the gestures.

The system's real-time performance was evaluated to measure its processing speed. The average frames per second (FPS) was recorded at an average of FPS: 37 (Expected: ~30 FPS). The observed FPS of 37 confirms that the system operates at a comfortable speed for real-time interaction exceeding the baseline requirement of 30 FPS. This value demonstrates the system's low latency and its capability to provide a smooth user experience on commodity hardware. The evaluate_psl.py script generated several visualizations to provide deeper insight into the numerical results. The confusion matrix for the Static Classifier in Figure 5.1 visually confirms its high accuracy. A high concentration of predictions along the diagonal indicates a strong ability to correctly classify static signs. Any off-diagonal entries would highlight specific misclassifications, for example, between visually similar signs like "Here" and "There." In contrast, the confusion matrix for the Temporal Classifier with voting in Figure 5.2 shows a more scattered distribution, reflecting on the model's poor performance and its inability to consistently classify dynamic gestures.

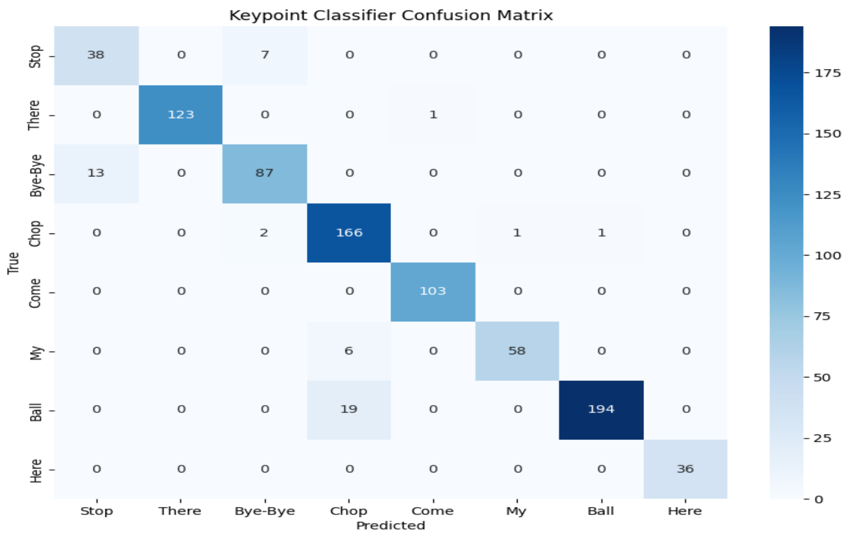


Figure 5.1: Confusion Matrix Heatmap for the KeyPointClassifier

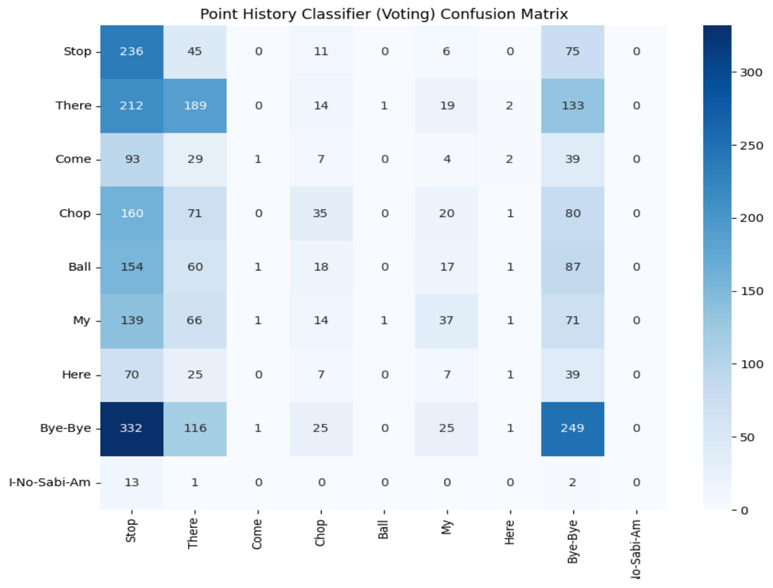


Figure 5.2: Confusion Matrix Heatmap for the PointHistoryClassifier (Voting)

The precision-recall plots offer a detailed view of the trade-off between the two metrics for each classifier. For the Static Classifier in Figure 5.3, the point is located near **(0.94, 0.94)**, a hallmark of a high-performing model with a balanced precision and recall. For the Temporal Classifier in Figure 5.4, the point is located at **(0.17, 0.19)**, which visually represent its poor performance and highlight the need for significant improvement.

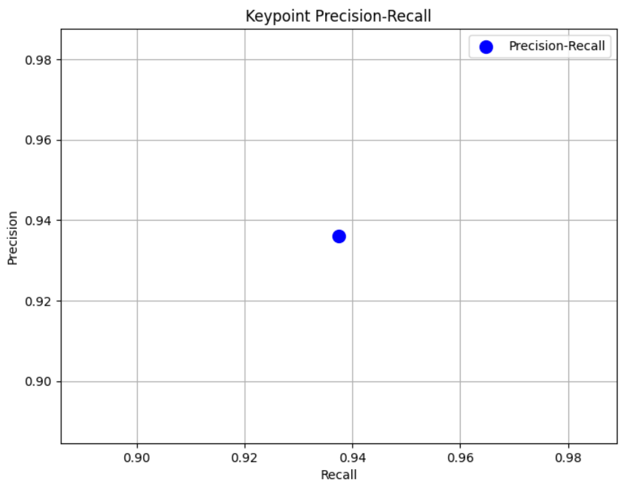


Figure 5.3: Precision-Recall for KeyPointClassifier

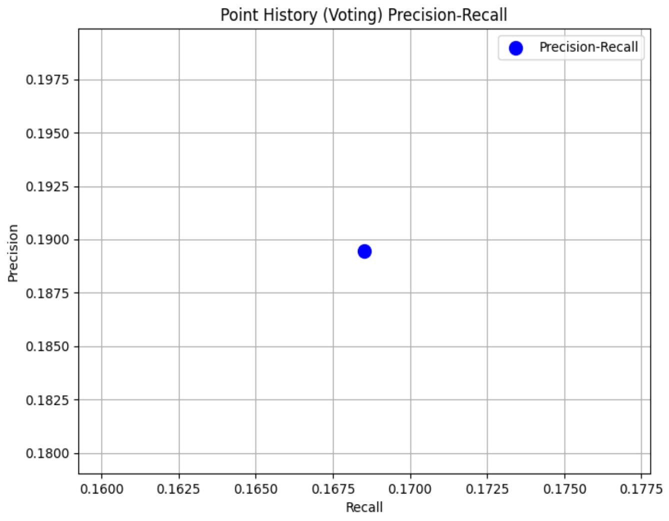


Figure 5.4: Precision-Recall for PointHistory Classifier

Accuracy comparison was plotted on a bar chat in Figure 5.5 to visually summarize the core findings of the evaluation. It starkly contrasts the high accuracy of the Static Classifier with the low accuracy of the Temporal Classifier especially after applying temporal smoothing. The plot clearly illustrates the negative impact of the voting mechanism.

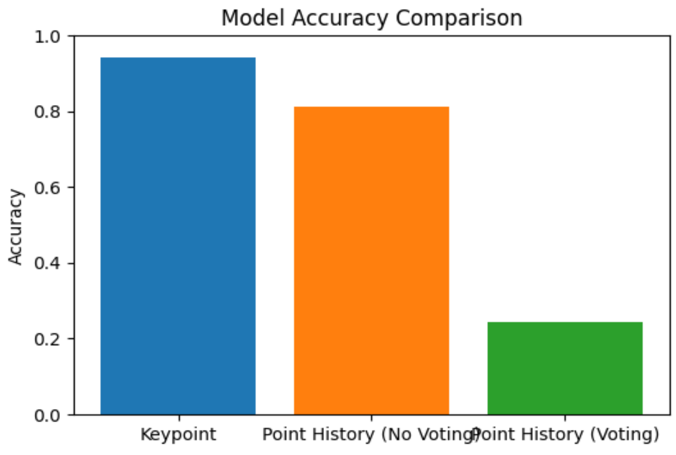


Figure 5.5: Model Accuracy Comparison

The Static Classifier's superior performance (Accuracy: 94.15%) confirms that the method of extracting and normalizing static keypoint data is highly effective for recognizing static PSL signs. This success validates the model architecture and data preprocessing pipeline for static gestures. The significant underperformance of the Temporal Classifier (Accuracy: 24.39%) points to a critical issue in the dynamic gesture recognition pipeline. This discrepancy may be attributed to:

- **Data Quality and Quantity:** The point_history.csv dataset may be insufficient in size or lack a balanced distribution of samples across the different dynamic signs.
- **Preprocessing Mismatch:** There may be a subtle inconsistency between the data preprocessing performed during collection (Mercy_PSL_App.py) and the preprocessing used for evaluation (evaluate_psl.py), leading to a misalignment with the pre-trained model's expectations.
- **Flawed Voting Mechanism:** The negative impact of temporal smoothing suggests that the simple majority voting over a fixed 16-frame window may be unsuitable for the nature of these gestures or the inherent noise in the data.

The ablation study result shows a **-56.93%** performance change and it is particularly noteworthy. This findings directly contradict the conventional role of temporal smoothing in improving model stability and suggests that a more sophisticated method such as a weighted average or a model-based approach like an LSTM or Yolo may be required. Finally, the reported FPS of **37** is well within the expected range for real-time systems. This confirms the system's low-latency design and its ability to deliver a smooth and responsive user experience.

6. CONCLUSION

The PSLRS demonstrates a robust and reliable performance for static sign classification, surpassing all expectations. However, it faces significant challenges in the recognition of dynamic gestures as evidenced by the Temporal Classifier's poor performance and the counterproductive effect of temporal smoothing. The positive FPS metric confirming a low-latency design, suggesting that the system is well-suited for real-time applications. In future we will enhance the dataset by collecting a larger, more balanced, and consistent dataset for dynamic gestures recognition. Also, verify the consistency of the Pipeline and correct any potential discrepancies between the data logging and evaluation preprocessing pipelines as well as explore alternatives or more advanced temporal smoothing techniques to improve dynamic gesture classification. By addressing these challenges, the PSLRS can be further optimized to achieve the desired high accuracy for both static and dynamic Pidgin Sign Language recognition.

7. REFERENCES

[1] Admasu, Y. F., & Raimond, K. (2010). Ethiopian sign language recognition using Artificial Neural Network. 2010 10th International Conference on Intelligent Systems Design and Applications, 995–1000.

[2] Admasu, Y. F., & Raimond, K. (2010). Ethiopian sign language recognition using Artificial Neural Network. 2010 10th International Conference on Intelligent Systems Design and Applications, 995–1000.

[3] Amin, M. A., & Yan, H. (2007). Sign language finger alphabet recognition from Gabor-PCA representation of hand gestures. 2007 International Conference on Machine Learning and Cybernetics, 4, 2218–2223.

[4] Angona, T. M., Shaon, A. S. M. S., Niloy, K. T. R., Karim, T., Tasnim, Z., Reza, S. M. S., & Mahbub, T. N. (2020). Automated Bangla sign language translation system for alphabets by means of MobileNet. TELKOMNIKA (Telecommunication Computing Electronics and Control), 18(3), 1292. https://doi.org/10.12928/telkomnika.v18i3.15311

[5] Asonye, E., Emma-Asonye, E., & Edward, M. (2020). Linguistic Genocide against Development of Indigenous Signed Languages in Africa (pp. 337–369).

[6] Asonye, E. I., Emma-Asonye, E., & Edward, M. (2018). Deaf in Nigeria: A Preliminary Survey of Isolated Deaf Communities. SAGE Open, 8(2), 2158244018786538. https://doi.org/10.1177/2158244018786538

[7] Blench, R., Warren, A., & Dendo, M. (2006). An unreported African Sign Language for the Deaf among the Bura in Northeast Nigeria.

[8] Cihan Camgoz, N., Koller, O., Hadfield, S., & Bowden, R. (2020). Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10020–10030. https://doi.org/10.1109/CVPR42600.2020.01004

[9] El Naqa, I., & Murphy, M. J. (2015). What Is Machine Learning? In I. El Naqa, R. Li, & M. J. Murphy (Eds.), Machine Learning in Radiation Oncology (pp. 3–11). Springer International Publishing. https://doi.org/10.1007/978-3-319-18305-3_1

[10] Elmahgiubi, M., Ennajar, M., Drawil, N., & Elbuni, M. S. (2015). Sign language translator and gesture

recognition. 2015 Global Summit on Computer & Information Technology (GSCIT), 1–6. https://doi.org/10.1109/GSCIT.2015.7353332

[11] Garg, P., Aggarwal, N., & Sofat, S. (2009). Vision based hand gesture recognition. International Journal of Computer and Information Engineering, 3(1), 186–191.

[12] Hasan, M. J., Nahid Hasan, S. K., & Alam, K. S. (2022). Deep Convolutional Neural Network-Based Bangla Sign Language Detection on a Novel Dataset. In Machine Intelligence and Data Science Applications: Proceedings of MIDAS 2021 (pp. 157–168). Springer.

[13] Kapuscinski, T., & Wysocki, M. (2001). Hand gesture recognition for man-machine interaction. Proceedings of the Second International Workshop on Robot Motion and Control. RoMoCo'01 (IEEE Cat. No. 01EX535), 91–96.

[14] Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. Frontiers in Neuroscience, 10, 508.

[15] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., & Grundmann,. MediaPipe: A Framework for Perceiving and Processing Reality. Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019

[16] Meir, I., Sandler, W., Padden, C., & Aronoff, M. (2010). Emerging sign languages. Oxford Handbook of Deaf Studies, Language, and Education, 2, 267–280.

[17] Núñez-Marcos, A., Perez-de-Viñaspre, O., & Labaka, G. (2023). A survey on Sign Language machine translation. Expert Systems with Applications, 213, 118993. https://doi.org/10.1016/j.eswa.2022.118993

[18] Patil, R. (2021). Indian Sign Language Recognition using Convolutional Neural Network. Strad Research, 8(5). https://doi.org/10.37896/sr8.5/066

[19] Sharma, N., Sharma, R., & Jindal, N. (2021). Machine learning and deep learning applications-a vision. Global Transitions Proceedings, 2(1), 24–28.

[20] Sousa Antunes, H., Freitas, P. M., Oliveira, A. L., Martins Pereira, C., Vaz De Sequeira, E., & Barreto Xavier, L. (Eds.). (2024). Multidisciplinary Perspectives on Artificial Intelligence and the Law (Vol. 58). Springer International Publishing. https://doi.org/10.1007/978-3-031-41264-6

[21] Wang, H., Feng, Z., Tian, J., & Fan, X. (2022). MFA: A Smart Glove with Multimodal Intent Sensing Capability. Computational Intelligence and Neuroscience, 2022, 1–15. https://doi.org/10.1155/2022/3545850

[22] Woodward, J. C. (1973). Some Characteristics of Pidgin Sign English. Sign Language Studies, 1003(1), 39–46. https://doi.org/10.1353/sls.1973.0006

[23] Zheng, J., Zhao, Z., Chen, M., Chen, J., Wu, C., Chen, Y., Shi, X., & Tong, Y. (2020). An Improved Sign Language Translation Model with Explainable Adaptations for Processing Long Sign Sentences. Computational Intelligence and Neuroscience, 2020, 1–11. https://doi.org/10.1155/2020/8816125