Signal, Image and Video Processing (2025) 19:682
https://doi.org/10.1007/s11760-025-04230-4

ORIGINAL PAPER

# Pakistan Sign Language Recognition: From Videos to Images

**Hafiz Muhammad Hamza**[1] · **Aamir Wali**[1]

Received: 5 February 2025 / Revised: 20 March 2025 / Accepted: 24 April 2025 / Published online: 2 June 2025


**Abstract**

Pakistan Sign Language (PSL) is the primary mode of communication for the Deaf community in Pakistan, yet research in PSL recognition remains underdeveloped. The primary challenge lies in the limitations of its dataset, the PSL Dictionary, which contains only two samples per sign, making it difficult to develop accurate recognition systems. While the action recognition models achieve high accuracy, they are resource-intensive, requiring substantial storage and computational power. To address these challenges, we propose a novel modality transformation approach that converts video-based sign language gestures into single images, allowing the use of image recognition models instead. In our approach, 21 key landmarks are first extracted from each video using MediaPipe. The movement of each landmark, such as the index fingertip, is tracked across all frames and represented as a single image per class, resulting in 21 distinct feature representations per video sample. To further enhance recognition performance, we introduce feature aggregation as part of this transformation. During feature aggregation multiple landmark images of the same class were combined into a single denser image. This integrated pipeline of modality transformation and feature aggregation allows the model to learn richer spatial representations from limited data. This reduces both storage and computational requirements while maintaining comparable performance to the action recognition models. We utilized Convolutional Neural Networks (CNN) to develop an efficient image recognition-based PSL recognition system. Although CNNs require a lot of training data, our model achieved superior performance in PSL recognition compared to existing techniques. Finally, building on our modality transformation approach, we also demonstrated the effectiveness of feature-specific ensemble learning, which achieved the best accuracy of 92.5% and an F1 score of 90%. Our comparative analysis highlights the effectiveness of the lightweight, cost-effective image recognition-based method for real-time, resource-constrained applications.

## 1 Introduction

In recent years, there has been growing interest in developing recognition systems for Pakistan Sign Language (PSL). To the best of our knowledge, there is only one notable previous work [1] that explored the application of action recognition models, including C3D, I3D, and TSM, to PSL recognition. These models demonstrated high accuracy in recognizing video-based sign gestures, establishing a robust baseline for PSL recognition systems. However, they also highlighted critical challenges that limit their broader applicability. These include significant storage and computational requirements, which hinder the development of lightweight and resource-efficient systems suitable for real-world applications.

A lot of work has been done on sign language recognition for other languages, with systems developed for American Sign Language (ASL) [2], Chinese Sign Language (CSL) [3], and Indian Sign Language (ISL) [4]. These studies have employed diverse methodologies, including deep learning approaches such as CNNs, Transformers [5], and Graph Neural Networks, to capture the intricate spatiotemporal features of sign gestures. For instance, ASL recognition systems have achieved significant advances in accuracy through large-scale datasets and 3D convolutional models, while CSL recognition has benefited from combining skeleton-based and video-based methods. Despite these advances in other languages, PSL recognition remains largely underexplored, with limited datasets and research. A singular but notable study on PSLR is [1], which utilized action recognition models and highlights several key challenges.

The PSL dictionary dataset, the primary resource for PSL recognition research, consists of video recordings of sign language gestures. Each sign is represented by only

✉ Aamir Wali
aamir.wali@nu.edu.pk

[1] FAST School of Computing, National University of Computer and Emerging Sciences, Lahore, Pakistan

two video samples, creating a severe limitation in terms of dataset size. Action recognition models used in [1] for PSLR, such as C3D, I3D, and TSM, also require substantial amounts of data to perform effectively. Additionally, action recognition models trained and tested on video datasets demand significant computational power, typically requiring a GPU for processing which may not be accessible for many researchers and practitioners. These challenges are further compounded when considering deployment on resource-constrained devices, where real-time processing is often required.

Due to these factors, developing a light-weight yet high-performing PSLR system becomes increasingly difficult, especially for deployment on devices with limited resources. The extensive storage needs make it challenging to collect or generate additional video data, while the high computational requirements limit the feasibility of cost-effective and real-time recognition systems.

To address these challenges, we propose a novel approach that eliminates dependency on the video dataset. Our strategy involves converting an entire video into a single image and then employing image recognition models to develop the PSL recognition system. This approach reduces computational overhead and storage requirements while achieving similar performance compared to that of action recognition models. After generating the image dataset, we utilize it to train several image recognition models, with the aim of building a lightweight PSLR system. Specifically, we focus on three models or techniques: CNN, SVM, and transfer learning. To enhance the performance of these models, we introduce different aggregation and ensembling techniques across various stages of the machine learning process.

The key contributions of this research are outlined below:

1. A novel approach is proposed to convert video data into image data. This reduces storage and computation requirements and removes dependencies like clothing styles, background colors, and lighting conditions.
2. Image recognition models are applied to the converted image data, delivering comparable results to action recognition models while requiring less storage and computation.
3. Novel ensembling strategies are proposed to enhance the performance of the PSLR system.

Building on our methodology, we conducted a number of experiments and provided an in-depth analysis of the results obtained. Our experiments demonstrate a notable improvement in the performance of the recognition models, reinforcing our objective of creating a lightweight and cost-effective recognition system for PSL. Additionally, we offer a comparative analysis of the two approaches developed for PSL recognition: one based on action recognition ( [1]) and the other on image recognition. This analysis shows that the latter approach achieves comparable performance while significantly reducing resource requirements.

This paper provides a comprehensive analysis of our proposed approach, detailing the methodology and experimental results. We also present a comparative analysis of action recognition and image recognition methods, demonstrating that the latter offers a viable alternative for real-time PSL recognition in resource-constrained settings. By addressing the limitations of previous approaches and leveraging innovative techniques, this work contributes to advancing PSL recognition research, paving the way for practical and efficient solutions.

The rest of the paper is organized as follows. Section 2 provides a brief literature review of related studies. Proposed methodology is given in Sections 3 and 4. Section 5 presents the results with conclusion in Section 6.

## 2 Literature review

Initial research in SLR explored various approaches to recognizing sign language gestures. Early methods relied on direct measurements using equipment such as gloves [6], which captured hand shape, finger positions, and movement trajectories. Sensors and motion-capturing devices like Kinect recorded depth information and dynamic motion patterns, enabling gesture recognition. These methods faced limitations in scalability and usability, prompting a shift to vision-based approaches. Vision-based methods capture signs via cameras and store them as images or videos. While image-based datasets [7] are limited to simple signs like alphabets and digits, video-based techniques offer broader applications by covering word-level and sentence-level sign language gestures.

One approach that is popular in the SLR community are the 2D Convolutional Neural Networks (2D CNNs). These are designed to process and analyze two-dimensional image data effectively. Their ability to generalize spatial hierarchies and adapt to complex visual patterns has also made them indispensable for SLR. For instance, [8] introduced an innovative deep learning pipeline for automated hand sign language recognition, combining the Single Shot Detector (SSD), 2D CNNs, 3D Convolutional Neural Networks, and Long Short-Term Memory (LSTM) networks, utilizing RGB video inputs. Beyond alphabet recognition, CNNs have been applied to recognize entire words and phrases in sign language. A study [9] utilized a CNN architecture to recognize Bangla Sign Language characters in real-time. The system translated recognized signs into corresponding textual Bangla characters, facilitating natural communication between deaf and hard-of-hearing individuals and the general population. The model achieved a validation accuracy

of 99.57%, demonstrating the potential of CNNs in real-time SLR applications.

To address the challenges posed by dynamic gestures, such as those involving motion, researchers have explored two-stream CNN architectures. One study [10] introduced a Two-Stream Mixed Convolutional Neural Network (TSM-CNN) for American Sign Language (ASL) recognition, which processes two consecutive images to capture temporal information. This approach improved the recognition of dynamic gestures involving letters.

Support Vector Machines (SVMs) have also been employed for SLR, particularly for classifying static hand gestures. [11] proposed a robust recognition system for Amharic Sign Language (AMSL) alphabets. In another study [12], the authors developed a system for recognizing Indonesian Sign Language (Bisindo) alphabets. Some researchers have tried combining CNNs with SVMs to improve the performance of the SLR system.

Researchers have employed a variety of data representation techniques to enhance the performance of sign language recognition systems. These approaches focus on transforming raw video and image data into meaningful representations that capture the essential features of hand gestures and movements. One such representation is offered by MediaPipe [13] that estimates 21 hand landmarks such as joint and finger tips coordinates from 2D RGB images, as illustrated in Figure 1. Studies such as [7] used MediaPipe for recognizing ASL characters for extracting landmarks from hand images captured by a standard webcam.

Recent advancements in activity recognition and ensemble learning have demonstrated the effectiveness of combining spatiotemporal features and multiple models for improved recognition accuracy. For instance, Tabish et al. [14] propose an activity recognition framework for sports videos, emphasizing the importance of capturing both spatial and temporal information in dynamic environments. This aligns with our approach of transforming video-based PSL gestures into spatial representations using hand landmarks. Similarly, [15] demonstrate the effectiveness of ensemble learning in real-time long-range object tracking, where multiple models are combined to improve robustness and accuracy. Their work supports our use of ensemble learning techniques, where multiple models trained on different landmark datasets are combined through majority voting to enhance PSL recognition performance.

### 2.1 Recognition of Asia's regional sign languages

Given the linguistic diversity in Asia, various regional sign languages have been the focus of SLR researchers. Among these, CSL has received the most attention, likely due to China's large population, which includes approximately 4 million sign language users. Additionally, government-backed research initiatives and the availability of extensive datasets have further contributed to its prominence in SLR studies. Similarly, Arabic Sign Language (ArSL), the primary mode of communication for the hearing-impaired community in more than 20 countries, has also been extensively studied. With over 400,000 users, ArSL has garnered significant research interest, driven by its widespread use across multiple nations. Similarly, research has been done for SIBI (Sistem Isyarat Bahasa Indonesia), Japanese, Korean, Bengali, etc. Some of these studies also employed Table 1 provides a summary of recent advancements in the recognition of regional sign languages across Asia.

### 2.2 Pakistan sign language recognition

PSL serves as the primary means of communication for the Deaf community in Pakistan. While a PSL recognition system is crucial and its dictionary dataset is publicly available [29], significant advancements have yet to be made in this area. Existing research largely relies on limited datasets that cover only alphabets, digits, or basic gestures. An overview of these recent developments is presented in Table 2.

A considerable number of researchers have favored SVMs for PSLR. Although [33] acquired input data from videos, their SLR technique was image based. They extracted four vision-based features, that is, local binary patterns, a histogram of oriented gradients, an edge-oriented histogram, and speeded-up robust features. These features were then classified using Multiple kernel learning (MKL) in SVM. Similarly, [36] also employed SVM for sign recognition. An SLR system for PSL static alphabet used Faster R-CNN for hand localization and symmetric mean-based binary patterns (sMBP) for feature extraction [40]. The signs were finally classified using error correction output codes (ECOC)-based SVMs. Additionally, [34] used the Speeded Up Robust Feature (SURF) method for feature extraction, clustering these features with K-means to create a Bag-of-Words (BoW) model, which was then classified using SVM.

Almost all research and development in PSL recognition has focused on image-based approaches or limited gesture sets, such as alphabet, digits, and emotions such as happy, sad, etc. Apart from [1], all other studies mainly cover static images. Few studies have explored different recognition algorithms for video datasets, but typically rely on relatively simple datasets. To the best of our knowledge, no study has leveraged the PSL Dictionary dataset to design an efficient and robust PSLR system capable of handling a large vocabulary.

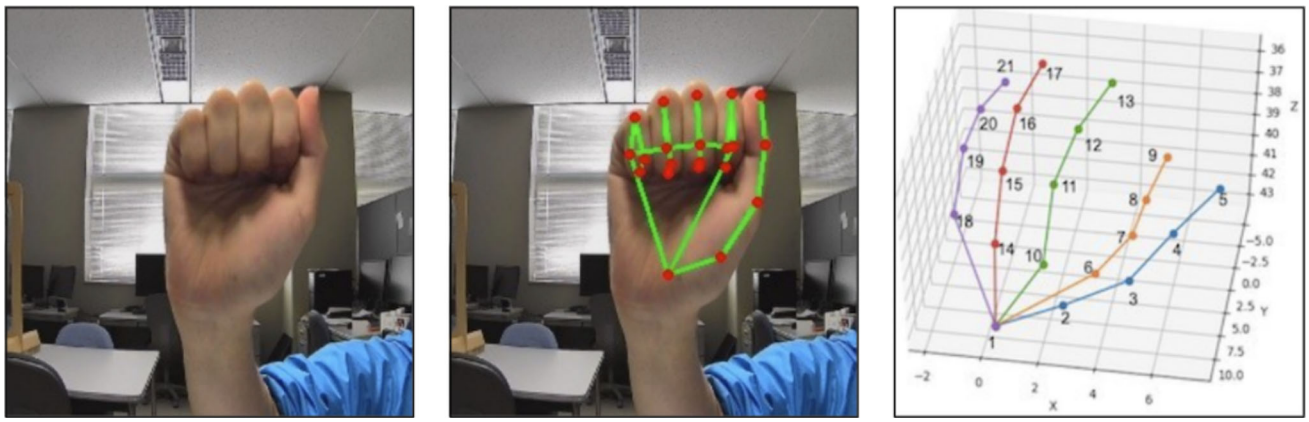


**Fig. 1** Extraction of 21 joint points using the MediaPipe API, as presented by Shin et al. [7]. The left image represents the input image, the middle image displays the estimated joints, and the right image illustrates the joint order

**Table 1** Overview of recent developments in the recognition of Asia's regional sign languages

| Sign Language | Technique | Acc. | Ref. |
|---|---|---|---|
| Arabic Sign Language | Transformation techniques and SVM | 99% | [16] |
| Arabic Sign Language | CNN, LSTM, and Self-MLP | 87.69% | [17] |
| Arabic Sign Language | CNN and MobileNet | 94.46% | [18] |
| Arabic Sign Language | CNN and Transfer learning | 99% | [19] |
| SIBI (Sistem Isyarat Bahasa Indonesia) | I3D and Transfer Learning | 97.50% | [20] |
| Japanese Sign Language | Hand pose estimation, Random forest (RF), and SVM | 98.40% | [21] |
| Korean Sign Language | Transformers and CNN | 98.30% | [22] |
| Korean Sign Language | CNN | 84.5% | [23] |
| Indian Sign Language | CNN | 99.90% | [24] |
| Indian Sign Language | CNN and SVM | 99.64% | [25] |
| Bengali Sign Language | CNN | 96.33% | [26] |
| Bengali Sign Language | CNN | 99.22% | [27] |
| Bengali Sign Language | CNN | 98.38% | [28] |

**Table 2** Comprehensive summary of research and development in the field of PSLR

| Technique | Dataset Classes | Dataset Type | Ref. |
|---|---|---|---|
| Artificial Neural Network (ANN) | Alphabet | Images | [30] |
| Template matching | Alphabet & digits | Images | [31] |
| K Nearest Neighbors (KNN) | Alphabet | Images | [32] |
| SVM | Alphabet | Images | [33] |
| SVM | Alphabet | Images | [34–36] |
| CNN | Digits | Images | [37] |
| CNN | Alphabet | Images | [38] |
| Graph NN | Gestures | Skeleton key points | [39] |
| I3D, TSM, C3D | PSL | Videos | [1] |

## 3 Proposed approach: transforming gesture videos to sign images

To convert video gestures into image data, we drew inspiration from the real-life process of drawing and painting. When a person sketches a digit, such as "8", their hand movement follows a specific trajectory. This specific movement generates the image of the digit. It can be said that the image is a function of hand movement. Hence, mathematically

$$Drawn\ Image = f(Hand\ Movement) \tag{1}$$

Analogously, our approach transforms dynamic video-based gestures into static image representations by capturing the essential movement patterns and encoding them as spatial features in a two-dimensional format. This transformation enables efficient recognition using image-based machine learning techniques while reducing computational complexity. Our methodology for generating image representations of PSL gestures involves the following steps:

1. **Hand landmark detection:** Detect a hand landmark (e.g., the index finger tip) in the video frames and track its movement across the entire video.
2. **Image conversion:** Plot this movement on a 2D image to generate *a* binary image corresponding to the gesture video, where the white pixels represent the landmark positions in different frames.

It is worth mentioning that the process of tracking landmarks and generating images requires minimal computational resources. Once the image data is generated, it can be used to train any image recognition machine learning model. This approach enables the development of a cost-effective PSL recognition system without concerns about storage limitations or high computational requirements. Not only does this reduce costs, but it also facilitates easy deployment on edge devices such as smartphones and tablets, enhancing day-to-day communication between the hearing impaired and able-bodied individuals in a more accessible and practical manner.

### 3.1 Benefits of converting videos to images

Transforming an entire video into a single image brings several notable benefits, making it a more practical and streamlined approach for specific applications.

- **Reduced dependency on signer:** By focusing on static images, dependencies on signer-specific features like gender, clothing, background, and lighting conditions are minimized. This creates a more uniform dataset that can be generalized across various settings.
- **Minimized impact of external factors:** By converting videos into image frames, the effects of varying external conditions, such as changing backgrounds, lighting, and shadows, are greatly reduced. These elements often introduce noise in video data, but static images can be curated to focus solely on the gesture or sign, ensuring that the model is not influenced by external visual inconsistencies or distractions. This improves the overall robustness and consistency of the data used for training.
- **Lower storage requirements:** Image data occupies significantly less storage compared to video files, making it easier to manage large datasets without the need for extensive storage solutions.
- **Reduced computational costs:** Processing static images requires less computational power compared to videos, particularly in tasks involving deep learning or machine learning. This reduction in computational cost makes training models more efficient and faster. This reduction also paves the way for developing low-cost sign language recognition systems that can be efficiently deployed on edge devices such as smartphones, tablets, and other IoT devices. These portable systems can significantly bridge the communication gap for the deaf and hard-of-hearing community, offering real-time accessibility and aiding in day-to-day interactions without the need for high-powered hardware.

### 3.2 Optimization techniques

To create an optimal dataset and achieve the best results, we experimented with different parameters for image generation, including:

- **Image size:** Determining the most effective dimensions for image representation.
- **Brush size:** Testing various brush sizes to identify the most distinguishable features.

Additionally, due to the limitations of the hand landmark detection algorithm, the generated images initially lacked continuous signs. To address this, we attempted to convert discontinuous signs into continuous representations to explore the potential for further improving system accuracy.

### 3.3 Transforming videos to images

To build an image recognition-based PSLR system, our first task was to convert the entire gesture from video into a single image. We utilized hand landmarks to achieve this conversion. For each video in our dataset, the following steps were executed:

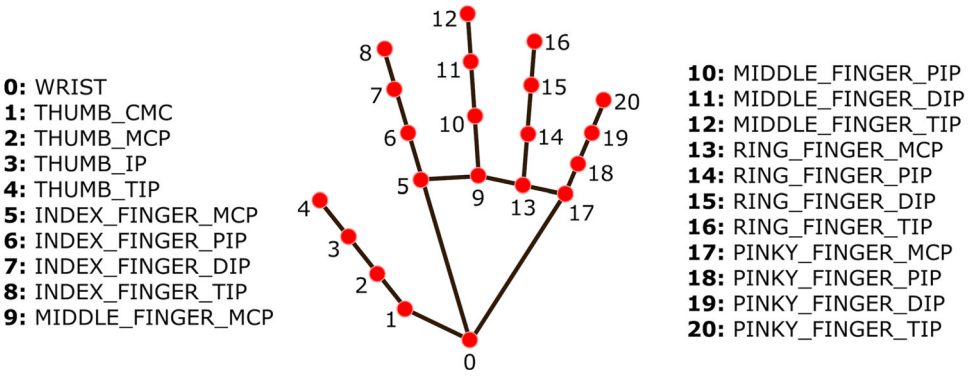



**Fig. 2** Landmarks detected by MediaPipe Hand Landmarker

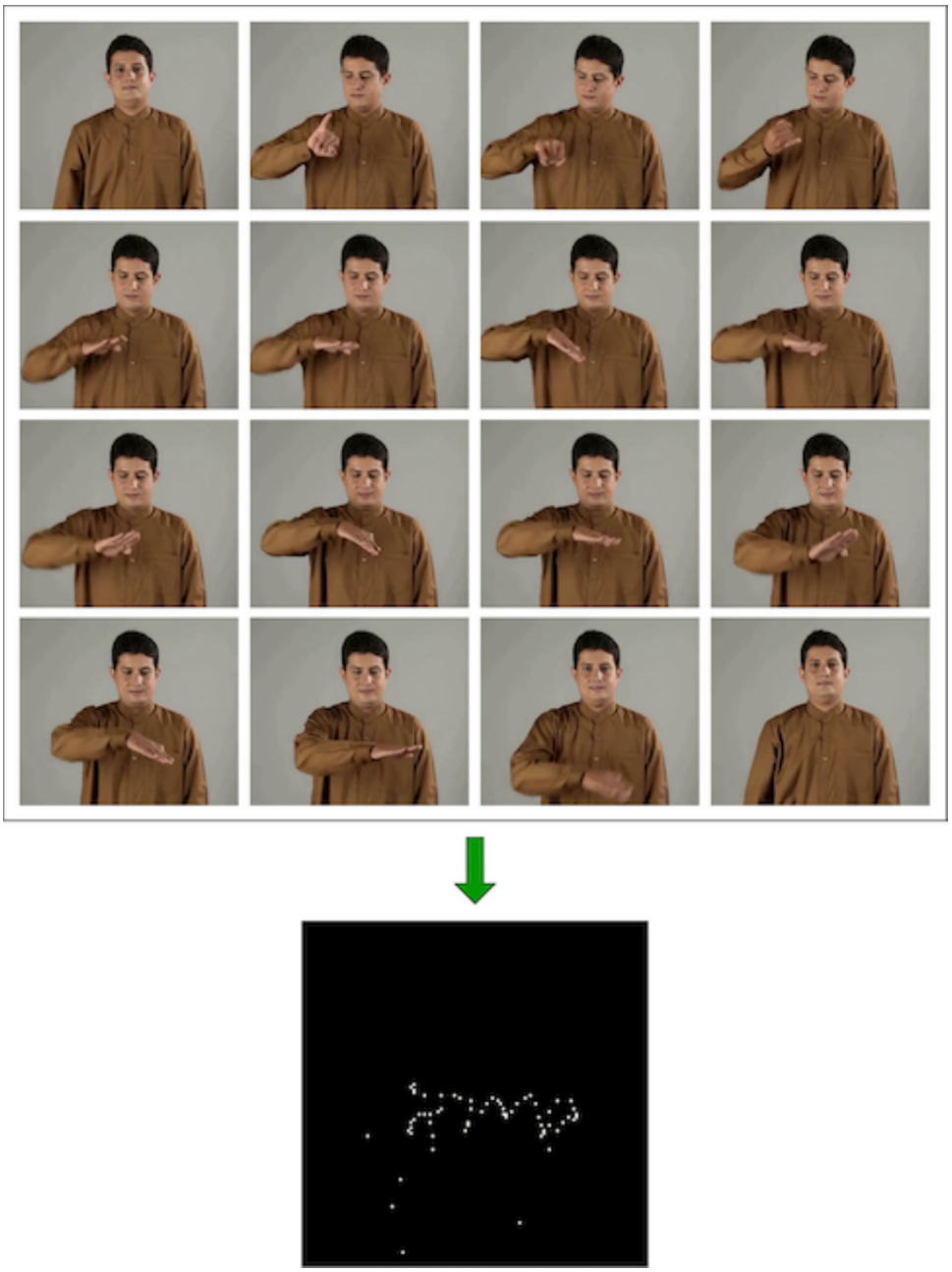


**Fig. 3** Transformation of video to image: the index finger tip of the signer is tracked in each frame and plotted on a 2D binary image

1. For each frame in the video, detect hand landmark(s) and store their coordinates in a list $V$.
2. Initialize an empty black image $I$.
3. For each coordinate in $V$, find it's mapping in $I$ and put a white pixel in that position.
4. Save $I$ as the image representation of the sign gesture.

Let $(x_v, y_v)$ be a coordinate from $V$ corresponding to a detected landmark. Let the width and height of the video frames are $W_v$ and $H_v$, respectively, and those of $I$ are $W_i$ and $H_i$, respectively. Then, the mapped coordinate in $I$ can be calculated as follows:

$$x_i = \frac{x_v}{W_v} \times W_i \quad , \quad y_i = \frac{y_v}{H_v} \times H_i \tag{2}$$

To demonstrate the implementation of the algorithm outlined above, we present a concise walk-through of its execution. First, for each frame in the video, we track the index finger tip and store it's coordinates in a list $V$. Subsequently, we initialize a 128 × 128 black image or matrix with pixel values set to zero. Then, we iterate $V$ and for each coordinate, we find its mapping in the image using Equation 2 and set it to white. Finally, the resultant image, corresponding to the respective sign gesture, is stored for further analysis and processing.

### 3.4 Hand Landmark Detection

We used MediaPipe Hand Landmarker task [13] to detect the landmarks of the hands. MediaPipe is an open-source framework developed by Google that offers a wide range of computer vision solutions, including hand landmark detection. MediaPipe Hand Landmarker task involves using deep learning models to accurately identify and track key points on a person's hand, allowing for real-time hand gesture recognition and precise hand movement analysis in various applications, such as augmented reality, sign language interpretation, and more. MediaPipe simplifies the development of these features by providing pre-trained models and easy-to-use APIs. MediaPipe Hand Landmarker detects 21 hand landmarks. These landmarks represent key points on the hand, including fingertips, knuckles, and wrist. The landmarks detected by MediaPipe and their corresponding values and descriptions are demonstrated in Figure 2.

Figure 3 highlights the application of the MediaPipe Hand Landmarker. First some of the frames extracted from a video depicting the sign gesture "Jeep" are shown. It can be seen that the signer forms a wavy pattern. In this particular example, we track the tip of the index finger in each frame and "paint" its position as white on a black background. The resultant image is also presented in Figure 3.

**Table 3** CNN performance on datasets with different image sizes

| Image size | Average accuracy | Best accuracy |
|---|---|---|
| 32 × 32 | 67.83 | 73.25 |
| 64 × 64 | 71.86 | 78.50 |
| 128 × 128 | 74.43 | 82.75 |
| 256 × 256 | 60.06 | 66.00 |

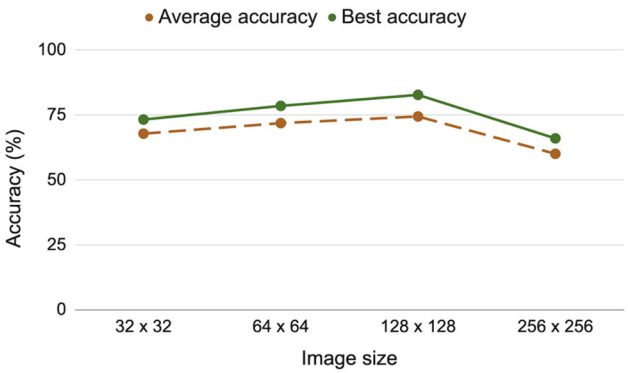



**Fig. 4** CNN performance on datasets with different image sizes

### 3.5 Optimal image settings

When transforming a video into an image, several settings and parameters must be considered, including image size and brush size. Initially, we discussed placing a white pixel on a black background to mark the landmark location. Instead of a single white pixel, a block of white pixels (e.g., 2 × 2) can be used. In addition, the pixel can have any color besides white. We experimented with various settings and analyzed their impact on classification accuracy to identify optimal values. To evaluate, we generated datasets with different settings and used a CNN, detailed in the next chapter. All experiments were conducted five times, with both average and best accuracies reported. We considered different image sizes with 128 × 128 image size producing the best results.

### 3.6 Image Size

We considered different image sizes. Our aim was to determine the smallest size of the image that generates the best results. The various image sizes tested and their corresponding accuracies are summarized in Table 3 and illustrated in Figure 4. It can be observed that the 128 × 128 image size produced the best results.

### 3.7 Brush Size

Each landmark detected in the video was mapped onto the image and represented as a white pixel or region. We experimented with different brush sizes: 1 × 1, 2 × 2, and 3 × 3.

**Table 4** CNN performance on images with different pixel sizes

| Pixel size | Average accuracy | Best accuracy |
|---|---|---|
| 1 × 1 | 74.43 | 82.75 |
| 2 × 2 | 73.96 | 80.75 |
| 3 × 3 | 71.83 | 77.5 |

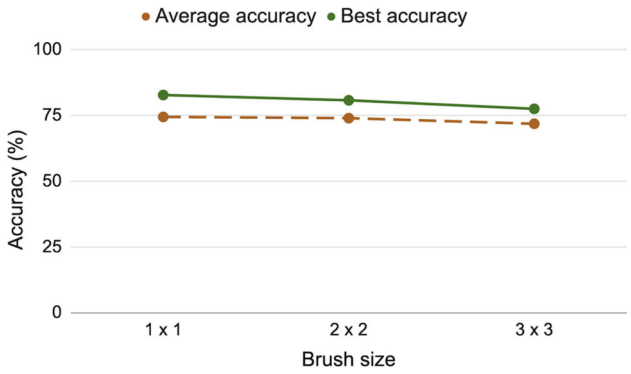



**Fig. 5** CNN performance on images with different brush sizes

The images with varying brush sizes were generated using the following strategy:

Given a coordinate $(x_i, y_i)$ that needs to be mapped onto the image $I$:

- The image with a $1 \times 1$ pixel size was generated by placing a white pixel at $I[x_i][y_i]$.
- The image with a $2 \times 2$ pixel size was generated by placing white pixels at $I[x_i][y_i]$, $I[x_i][y_i + 1]$, $I[x_i + 1][y_i]$, and $I[x_i + 1][y_i + 1]$.
- The image with a $3 \times 3$ pixel size was generated by placing white pixels at $I[x_i - 1][y_i - 1]$, $I[x_i - 1][y_i]$, $I[x_i - 1][y_i + 1]$, $I[x_i][y_i - 1]$, $I[x_i][y_i]$, $I[x_i][y_i + 1]$, $I[x_i + 1][y_i - 1]$, $I[x_i + 1][y_i]$, and $I[x_i + 1][y_i + 1]$.

The accuracies achieved by these different datasets containing images with varying pixel sizes are depicted in in Table 4 and Figure 5. It can clearly be seen that the $1 \times 1$ pixel size is the most effective for generating the PSL image dataset. Larger brush sizes ($2 \times 2$ and $3 \times 3$) led to reduced classification performance, likely due to the dilution of precise spatial information. A $1 \times 1$ pixel retains the fine-grained trajectory details required for recognition. Moreover, increasing image resolution beyond $128 \times 128$ did not improve accuracy, indicating that higher resolutions might introduce redundant spatial noise.

## 3.8 Pixel Colors

After determining the optimal dimensions for image size and pixel size, we experimented with different pixel colors to complete the process of optimizing the dataset. It's worth mentioning that the background of all generated images was Black (0, 0, 0). The pixel colors tested were White (255, 255, 255), Red (255, 0, 0), Green (0, 255, 0), Blue (0, 0, 255), Yellow (255, 255, 0), Fuchsia (255, 0, 255), and Aqua (0, 255, 255). These colors were chosen by maximizing the value (255) in each of the RGB color channels.

**Table 5** CNN performance on images with different pixel colors

| Color | Best accuracy |
|---|---|
| Blue (0, 0, 255) | 71.75 |
| Fuchsia (255, 0, 255) | 77.75 |
| Red (255, 0, 0) | 78.75 |
| Aqua (0, 255, 255) | 78.75 |
| Green (0, 255, 0) | 80.00 |
| Yellow (255, 255, 0) | 81.25 |
| White (255, 255, 255) | 82.75 |

The accuracies obtained with these different colors are presented in sorted order in Table 5. It can be observed that accuracy improved as the colors became brighter. This is intuitive because a brighter pixel on a black background is more likely to be distinguished and effectively trained by the CNN. Another observation is that the green channel appears to be the most effective, as accuracies improved when the green channel was set to 255.

## 3.9 Augmentation

Augmentation techniques are widely employed to expand training datasets and enhance model accuracy. Various augmentation methods exist for images, including noise addition, brightness adjustment, geometric transformations, oversampling, and generative approaches. These techniques have been successfully applied across diverse image domains, such as natural images [41], medical imaging [42], and handwritten text [43], demonstrating significant improvements in classification performance.

Building on the effectiveness of augmentation techniques demonstrated in the previous work [1], we explored various image augmentation strategies to enhance the performance of our PSL recognition system. Below are the details of the augmentation techniques we considered and their impact on model performance.

- **Brightness:** Brightness augmentation, commonly used for improving model robustness in grayscale or RGB images, was not applicable to our dataset. Unlike continuous pixel values in standard images, our dataset consists of strictly binary images (0 or 1 pixel values). Since brightness adjustments modify pixel intensity, they have

no effect on binary images, making this technique unsuitable for our use case.

- **Noise:** In conventional image processing, noise is introduced to help models generalize better by making them more robust to variations. However, in our dataset, hand landmarks are represented as discrete points (dots/pixels). Introducing noise could create additional unintended pixels that might be misinterpreted as landmarks, leading to incorrect feature representations and reduced model accuracy.
- **Rotation:** We applied rotation augmentation by rotating each image by a 10° angle in both directions:
  - $\mathbf{R}_1$: Each image was rotated 10° counterclockwise.
  - $\mathbf{R}_2$: Each image was rotated 10° clockwise.
- **Translation:** We applied translation augmentation by shifting each image by 10 pixels along the x-axis:
  - $\mathbf{T}_1$: Each image was translated $+10$ pixels along the x-axis.
  - $\mathbf{T}_2$: Each image was translated $-10$ pixels along the x-axis.

The results of the CNN evaluation on these augmented datasets are summarized in Table 6. To provide a comparative analysis, we also included the baseline accuracy obtained on the original dataset (denoted as "O").

Interestingly, our findings revealed that augmentation techniques not only failed to enhance model performance but instead led to a decline in accuracy. While both rotation and translation resulted in lower accuracy, translation had the most detrimental effect. Specifically, adding $T_1$ to the training dataset caused a 20% accuracy drop, and introducing $T_2$ led to a further 7% decrease. These results suggest that shifting landmarks spatially disrupts the model's ability to learn meaningful gesture representations, thereby reducing classification performance.

Generally, augmentation techniques, such as rotation, translation, noise, etc. can sometimes be ineffective in systems that rely on spatial trajectory-based representations rather than raw image pixels. This limitation has been noted in various studies, including those focusing on geometric transformations [8, 44] and noise-based augmentations [45]. These findings align with our results and observation that augmentation techniques led to a significant drop in accuracy, reinforcing the need for alternative approaches like feature aggregation and ensemble learning, which preserve the spatial relationships between landmarks.

Thus, while augmentation is often beneficial in traditional image-based learning tasks, our results indicate that for PSL recognition using hand landmark images, geometric augmentations may introduce more variability than the model can handle, leading to suboptimal results.

### 3.10 Discontinuous signs

It can be seen in Figure 3 that the sign we get is not a continuous trail. Rather, it's a discontinuous line containing many gaps among points. This is caused due to the limitation of MediaPipe Hand Landmarker task. Rapid motion or motion blur in the video frames can make it difficult for MediaPipe to track landmarks accurately. However, this is beyond the scope of our research work.

Let a gesture video be represented as a sequence of frames:

$$V = \{F_1, F_2, ..., F_N\}$$

where each frame $F_i$ contains a set of hand landmark points detected by MediaPipe:

$$F_i = \{p_1, p_2, ..., p_k\}$$

where $p_j = (x_i, y_i)$ represents the 2D coordinates of the $j^{th}$ landmark.

Due to limitations in MediaPipe tracking, some landmarks may be missing in certain frames, leading to discontinuities in the trajectory. Let $M \subseteq F_i$ denote the set of missing points in a given frame. If $|M| > 0$, then the trajectory is incomplete, forming gaps in the reconstructed gesture.

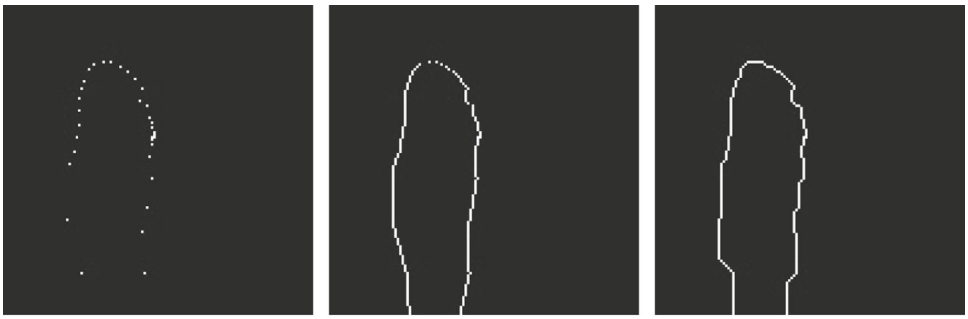
We hypothesized that discontinuities in irregular signs, caused by missing points, could negatively impact recognition accuracy. This assumption was based on the idea that incomplete or fragmented representations of hand gestures might lead to loss of critical structural information. Since sign language recognition relies on the spatial and temporal relationships between key points, gaps in the landmark data could disrupt the model's ability to learn meaningful patterns. As a result, the recognizer might misinterpret gestures or fail to differentiate between similar signs, leading to decreased accuracy. Furthermore, we assumed that because a gesture video represents a continuous sign where hand landmarks are visible in every frame, the same continuity should be preserved in the generated images.

To address this issue, we attempted to fill the gaps and generate continuous sign representations. Our first approach utilized the linear interpolation method, where we applied the equation-of-line algorithm (Algorithm 1) to estimate the missing points and construct a continuous trajectory. The resultant images for selected signs are shown in Figure 6 (b) and (e).

After testing this dataset, we observed that the accuracy of our recognizer decreased instead of increasing. To overcome this problem, we designed another algorithm (Algorithm 2) that progressively fills the gaps, i.e., the algorithm considers each point and draws a path in the direction of the next point. This approach is similar to one used in RRT [46] to connect a new node to the existing tree. A comparison of the accu-

**Table 6** CNN performance on different augmented datasets

| Dataset | Samples per class | Total samples | Accuracy |
|---|---|---|---|
| O | 1 | 80 | 82.75 |
| O + N | 2 | 160 | 66.25 |
| O + $R_1$ | 2 | 160 | 70.50 |
| O + $R_1$ + $R_2$ | 3 | 240 | 76.00 |
| O + $T_1$ | 2 | 160 | 62.75 |
| O + $T_1$ + $T_2$ | 3 | 240 | 55.75 |
| O + $R_1$ + $T_1$ | 3 | 240 | 64.25 |
| O + $R_1$ + $R_2$ + $T_1$ + $T_2$ | 5 | 400 | 63.25 |



(a) Off (Original) (b) Off (Algorithm 1) (c) Off (Algorithm 2)

(d) Annual (Original) (e) Annual (Algorithm 1) (f) Annual (Algorithm 2)

**Fig. 6** A comparison of the discontinuous and continuous signs

**Algorithm 1** Adding missing points using equation-of-line algorithm

```
function AddMissingPoints(points)
    for i ← 0 to points.length do
        x1 ← points[i][0]          y1 ← points[i][1]
        x2 ← points[i + 1][0]      y2 ← points[i + 1][1]
        a ← y2 − y1
        b ← x1 − x2
        c ← (a ∗ x1) + (b ∗ y1)
        for y ← y1 to y2 do
            x ← (c − b ∗ y) / a
            points[i + 1].Insert([x, y])
        end for
    end for
end function
```

racies achieved using continuous signs (generated using the two algorithms) and discontinuous signs (from the original dataset) is presented in Table 7.

**Table 7** CNN performance on continuous and discontinuous signs

| Dataset | Average accuracy | Best accuracy |
|---|---|---|
| Original | 74.43 | 82.75 |
| Algorithm 1 | 66.51 | 72.75 |
| Algorithm 2 | 67.37 | 74.50 |

The experiments revealed that the recognizer performed better with discontinuous signs compared to the continuous ones generated using the two algorithms. This discrepancy might be attributed to how discontinuous signs preserve landmark positions that the classification model learns as distinctive features. However, when these landmarks were connected into continuous lines, critical distinguishing features between different signs were lost or distorted, negatively impacting recognition accuracy.

**Algorithm 2** Adding missing points in a progressive manner

```
function AddMissingPoints(points)
    for i ← 0 to points.length do
        x1 ← points[i][0]        y1 ← points[i][1]
        x2 ← points[i + 1][0]    y2 ← points[i + 1][1]
        if ManhattanDistance(x1, y1, x2, y2) > 1 then
            x, y ← x1, y1
            if x1 < x2 then
                x ← x1 + 1
            else
                if x1 > x2 then
                    x ← x1 − 1
                end if
            end if
            if y1 < y2 then
                y ← y1 + 1
            else
                if y1 > y2 then
                    y ← y1 − 1
                end if
            end if
            points[i + 1].Insert([x, y])
        end if
    end for
end function
```

For instance, consider the following two sets of points representing pixels in two different discontinuous signs:

$$S_1 = \{(1, 1), (3, 3), (4, 4), (6, 6), (7, 7), (9, 9)\}$$
$$S_2 = \{(1, 1), (2, 2), (4, 4), (5, 5), (8, 8), (9, 9)\}$$

In their original discontinuous form, these sets contain distinct spatial patterns that the model can use to differentiate between signs. However, when converted into continuous trajectories, both sets will form identical lines from (1, 1) to (9, 9). As a result, the unique features that previously helped distinguish one sign from another are diminished, making it more challenging for the classifier to differentiate between them.

### 3.11 Limitation

One limitation of our research work is that the PSL signs used in our study were limited to those performed with one hand only. This restriction arises because MediaPipe detects landmarks without distinguishing between the right and left hand. For instance, when it identifies an index fingertip, it does not specify whether it belongs to the right or left hand; it simply indicates that an index fingertip has been detected. This lack of distinction caused confusion in our algorithms for handling missing points.

### 3.12 Conclusion

Based on our experimentation, we concluded that the optimal dataset for our image-based PSL recognition system consists of images with a size of 128 × 128, where each landmark is represented by a 1 × 1 white pixel. Furthermore, the use of continuous signs reduced the performance of the recognition models.

## 4 Methodology

### 4.1 Dataset

For this study, we used the Pakistan Sign Language Dataset available at www.psl.org.pk. This dataset is one of the primary resources available for PSL recognition research and consists of video recordings of various sign gestures. Given the limited availability of large-scale PSL datasets, we selected 80 videos corresponding to 80 different signs, ensuring that each sign is represented by a single video.

The PSL dataset used in this study is derived from the PSL Dictionary dataset, which contains signs for over 6000 words. However, for this research, we selected 80 frequently used words from the dataset. The dataset includes videos where each word is signed twice in a single video. As a consequence, the main challenge of using this dataset is that there are only two samples per word/class. One sample must be used for training while the other sample must be used for testing. Both male and female signers were employed by the dataset creators. The videos also display the signed word in Urdu and English, with some cases including supporting images.

The PSL dataset used in this study has also been employed in previous research, notably in *Pakistan Sign Language Recognition: Leveraging Deep Learning Models with Limited Dataset* [1]. The previous study explored video-to-image transformation techniques for PSL recognition and applied data augmentation to improve model performance. Our work builds upon this foundation by utilizing the original video data directly for sign recognition using deep learning techniques.

By using this dataset, we aim to contribute to the ongoing efforts in PSL recognition and facilitate the development of more robust and efficient sign language recognition systems tailored to the needs of the Deaf community in Pakistan.

### 4.2 Generation of Image Dataset from PSL Videos

We utilized the MediaPipe Hand Landmarker to detect 21 hand landmarks, which served as features for generating multiple datasets. For instance, when transforming a video into an image using the tip of the index finger, we tracked the INDEX_FINGER_TIP (8) landmark across all frames and recorded its movement in the form of an image. Since we had 80 sign gestures or classes in our dataset, this process was repeated for all classes resulting in 80 images. All of these

**Table 8** A summary of the datasets corresponding to each hand landmark, where each dataset consists of 80 images.

| Landmark | Train dataset | Test dataset |
|---|---|---|
| WRIST (0) | ds_0_tr | ds_0_ts |
| THUMB_CMC (1) | ds_1_tr | ds_1_ts |
| THUMB_MCP (2) | ds_2_tr | ds_2_ts |
| THUMB_IP (3) | ds_3_tr | ds_3_ts |
| THUMB_TIP (4) | ds_4_tr | ds_4_ts |
| INDEX_FINGER_MCP (5) | ds_5_tr | ds_5_ts |
| INDEX_FINGER_PIP (6) | ds_6_tr | ds_6_ts |
| INDEX_FINGER_DIP (7) | ds_7_tr | ds_7_ts |
| INDEX_FINGER_TIP (8) | ds_8_tr | ds_8_ts |
| MIDDLE_FINGER_MCP (9) | ds_9_tr | ds_9_ts |
| MIDDLE_FINGER_PIP (10) | ds_10_tr | ds_10_ts |
| MIDDLE_FINGER_DIP (11) | ds_11_tr | ds_11_ts |
| MIDDLE_FINGER_TIP (12) | ds_12_tr | ds_12_ts |
| RING_FINGER_MCP (13) | ds_13_tr | ds_13_ts |
| RING_FINGER_PIP (14) | ds_14_tr | ds_14_ts |
| RING_FINGER_DIP (15) | ds_15_tr | ds_15_ts |
| RING_FINGER_TIP (16) | ds_16_tr | ds_16_ts |
| PINKY_FINGER_MCP (17) | ds_17_tr | ds_17_ts |
| PINKY_FINGER_PIP (18) | ds_18_tr | ds_18_ts |
| PINKY_FINGER_DIP (19) | ds_19_tr | ds_19_ts |
| PINKY_FINGER_TIP (20) | ds_20_tr | ds_20_ts |

80 images that correspond to the INDEX_FINGER_TIP (8) landmark formed the index finger tip dataset. When this process is repeated for all 21 landmarks, we will ultimately get 21 distinct datasets.

The main benefit of generating multiple datasets is the increased training data size, as we originally had only *one* sample per class in the training set. Additionally, evaluating multiple landmarks offers insights into the most critical landmarks for image recognition models.

Recall that for every sign gesture in the PSL dataset, the signer performs the gesture twice. We used one instance for training and the other for testing. The process of generating 21 image datasets was applied to both training and testing videos.

To differentiate between them, we appended "tr" and "ts" to the names of the training and testing datasets, respectively. Additionally, the integer ID of each landmark was included in the dataset name for easy identification. The summary of these datasets can be found in Table 8.

## 4.3 Classification

With reference to the literature review, we identified the state-of-the-art image recognition techniques and applied them to our image recognition-based PSLR system. Specifically, we utilized CNN, SVM, and transfer learning. Each of these models were trained and tested on each of the 21 datasets. These models and techniques are discussed in detail in this section, along with their architecture and implementation.

### 4.3.1 Convolution neural network (CNN)

CNNs are types of deep learning models that have become the standard for tasks involving image recognition. Unlike traditional methods that require manual feature extraction, CNNs automatically learn to identify important features directly from the raw image data. They do this through a series of layers that gradually transform the input images into a set of high-level features, making it easier to classify or detect objects within the images.

In building our CNN model, we incorporated a series of convolutional and pooling layers to effectively extract and downsample features from the input images. The architecture begins with a rescaling layer to normalize the pixel values, followed by five convolutional layers with increasing filter sizes (16, 32, 64, 128, 256) and ReLU activation functions to capture intricate patterns within the images. MaxPooling2D layers are utilized to reduce the spatial dimensions of the feature maps, focusing on the most salient features. The output is then flattened into a vector and passed through a dense layer of 512 neurons before the final classification layer. This architecture is illustrated in Figure 7 and the details about each layer are presented in Table 9.

We utilized the Keras `Sequential` model for image classification to build our CNN model. We experimented with different optimizers (Adam, Adamax, Nadam, and RMSprop) to find the optimal one for our system and ultimately compiled our model with Nadam.

### 4.3.2 Support vector machine (SVM)

SVM is another machine learning algorithm used primarily for classification tasks. The main idea behind SVM is to find a line (or hyperplane in higher dimensions) that best separates the different classes of data. The algorithm aims to maximize the margin between the closest data points (known as support vectors) of each class to ensure that the model generalizes well to unseen data.

In order to develop our SVM model, we utilized the Support Vector Classifier (SVC) from the scikit-learn library. To find the optimal configuration, we utilized an approach similar to Grid Search where we rigorously experimented with various combinations of parameters, including different values of $C$ (0.1, 1, 10, 100), kernel types ('linear', 'rbf', 'polynomial', 'sigmoid'), and gamma values ('scale', 'auto', 1, 0.1, 0.01, 0.001). After thorough testing and evaluation, we determined that the optimal parameters for our model were `C=0.1`, polynomial kernel (`kernel = 'poly'`),

**Table 9** Summary of our CNN model

| Layer | Filters | Activation Function | Number of Parameters |
|---|---|---|---|
| Rescaling | - | - | - |
| Conv2D | 16 | ReLU | 448 |
| MaxPooling2D | - | - | - |
| Conv2D | 32 | ReLU | 4,640 |
| MaxPooling2D | - | - | - |
| Conv2D | 64 | ReLU | 18,496 |
| MaxPooling2D | - | - | - |
| Conv2D | 128 | ReLU | 73,856 |
| MaxPooling2D | - | - | - |
| Conv2D | 256 | ReLU | 295,168 |
| MaxPooling2D | - | - | - |
| Flatten | - | - | - |
| Dense | 512 | Sigmoid | 2,097,664 |
| Dense | 80 | - | 41,040 |

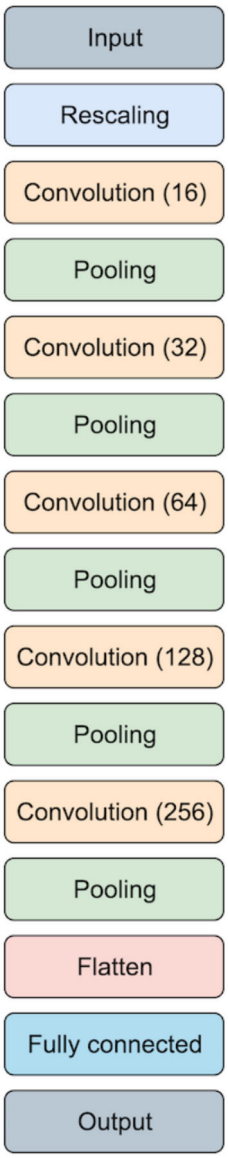



**Fig. 7** Convolution Neural Network (CNN) architecture

and `gamma = 'scale'`. To ensure that all features contributed equally to the model's decision boundary, feature scaling was applied using `StandardScaler`, which normalizes the features in the dataset to have a mean of 0 and a standard deviation of 1.

#### 4.3.3 Transfer learning (TL)

Another classification technique we adopted for developing our image-based PSLR system is transfer learning (TL). In many real-world applications, obtaining a large labeled dataset is impractical. Transfer learning [47] addresses this challenge by leveraging knowledge from pre-trained models on large datasets to improve performance on more specific tasks. It is a versatile methodology rather than a standalone model, as it can be integrated into various deep learning architectures, including CNNs. Transfer learning exploits the knowledge acquired by a pre-trained model on a large dataset and applies it to a related task with a smaller dataset, thereby improving performance and reducing the training time. Transfer learning has been used for all kinds of data including images [48], text [49] and audio. Within SLR, it has also been utilized in studies such as [50, 51] that explored various CNN architectures like AlexNet, VGG16, and VGG19 for robust feature extraction in tasks such as object recognition and image classification

In this study, we used the InceptionV3 model, pre-trained on the ImageNet dataset, as the base model for our classification task.

To adapt the InceptionV3 model for PSL recognition, we froze the majority of its layers, allowing only the top 30 layers (out of the 311 layers) to be trainable. This strategy retained the model's ability to extract meaningful features from images while fine-tuning the upper layers to better cap-

ture the details of the sign gestures. On top of the base model, we added a custom classification head consisting of a Global Average Pooling layer, a fully connected Dense layer with 512 units and ReLU activation, a Dropout layer for regularization, and a final Dense layer with a sigmoid activation function to classify the images into the desired number of classes.

### 4.4 Feature aggregation

Feature aggregation is a technique in machine learning where multiple features or representations, often derived from different sources or methods, are combined into a single, unified feature set. This process enhances the richness and comprehensiveness of the data provided to a machine learning model, thereby improving its ability to learn complex patterns. The rationale behind feature aggregation is that different features may capture distinct aspects of the underlying data, and by combining them, the model can achieve a more holistic understanding of the problem at hand. Aggregation can be done in several ways, such as simple concatenation of feature vectors, where features from different domains are merged, or through summarization techniques like statistical pooling, where operations such as mean, median, or max pooling are applied across time-series or spatial dimensions to create more meaningful representations.

As discussed earlier, the MediaPipe Hand Landmarker detects 21 hand landmarks. Using these hand landmarks, we generated multiple training and corresponding testing datasets, as explained in section 4.2. Initially, we tested our models on these individual datasets to identify the top-performing $\mathscr{L}$ landmarks. This experiment revealed the best landmarks for each model. With the assumption that a classification model trained on a few top landmarks could produce better results compared to a model trained on any single landmark, we employed feature aggregation and dataset consolidation to combine the selected landmarks effectively.

In our case, we combined multiple landmarks into a single sample to generate denser images containing data from multiple landmarks. For instance, top $\mathscr{L}$ landmarks were tracked in each video, and the coordinates of all $\mathscr{L}$ landmarks were mapped onto a single image to create the feature aggreagation dataset. In this study, we set $\mathscr{L} = 5$, focusing on the top five landmarks. Figure 8 illustrates a few samples generated using feature aggregation with $\mathscr{L} = 5$. To evaluate the models trained on the feature aggregation dataset, corresponding test samples were generated using the same strategy. Since both the training and testing datasets contained one sample per class, the models were evaluated using the standard machine learning evaluation strategy.

### 4.5 Dataset consolidation

Building on our feature aggregation technique, we employed a dataset consolidation approach, where the datasets corresponding to the top-performing $\mathscr{L}$ landmarks were grouped together to form a larger dataset. In this approach, each class contained $\mathscr{L}$ samples, effectively addressing the limitation of having only one sample per class in the original dataset. However, testing the models trained on a consolidated dataset posed an intriguing research challenge, which is discussed in the next section.

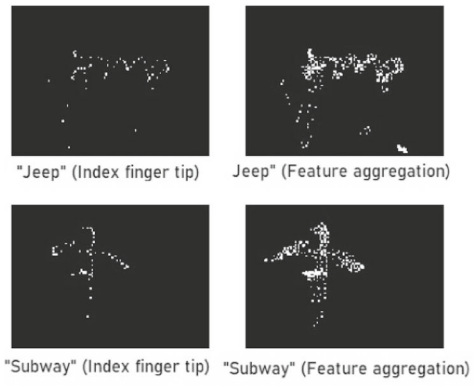
#### 4.5.1 Testing methodology for consolidated dataset

The dataset consolidation technique produces a larger dataset by combining the datasets of the $\mathscr{L}$ top-performing landmarks. As a result, each class contains $\mathscr{L}$ samples, with each sample representing the movement of a different landmark. This consolidated dataset is then used to train a classifier. However, a critical question arises: which of these $\mathscr{L}$ samples should be used to test the model?

Consider a PSLR system that has been developed and trained using the consolidated dataset. In practical scenarios, during inference, the system takes a sign gesture video as input and outputs the corresponding word represented by that gesture. From the video, $\mathscr{L}$ images of the top-performing landmarks are generated. During testing, a key challenge is deciding which of these $\mathscr{L}$ images (representing $\mathscr{L}$ different landmarks) should be used to evaluate the model. This consideration is vital because the recognition system was trained on the consolidated dataset, which represents multiple landmarks. To address this issue, several testing strategies were considered as discussed below:

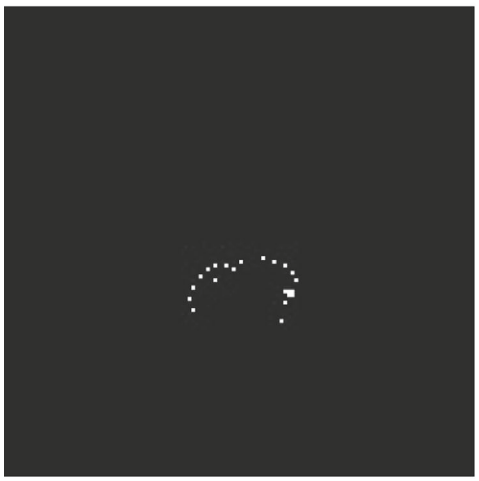
1. **Random landmark**: Randomly select one landmark from the 5 landmarks to test the model's performance.
2. **Best-performing landmark**: Test the model on the landmark that showed the highest accuracy among the 5 landmarks.
3. **Few top landmarks**: Select the top few landmarks from the 5 landmarks and test the model on these.
4. **All landmarks**: Evaluate the model on all 5 landmarks.

Testing the model on a single landmark may not yield the most accurate results, as it can be biased toward that specific landmark and fail to provide a comprehensive evaluation of the model's performance. A similar issue arises when testing the model on only a subset of the top landmarks. For example, if a model is trained on five landmarks but tested on only one or a few, it risks overfitting or underfitting to those specific features, leading to biased and inaccurate outcomes. Inspired by concepts such as crowd aggregation and collective intelligence, a more effective approach is to test

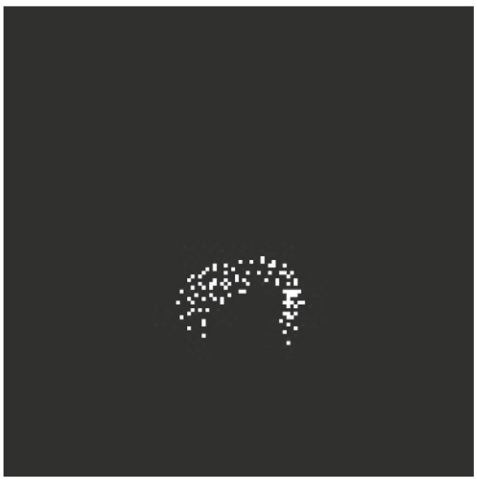



(a) "Jeep" (Index finger tip)

(b) "Jeep" (Feature aggregation)



(c) "I" (Index finger tip)



(d) "I" (Feature aggregation)

(e) "Subway" (Index finger tip)

(f) "Subway" (Feature aggregation)

**Fig. 8** Comparison of selected samples from the original dataset with their feature-aggregated counterparts, where five landmarks are tracked instead of one in feature aggregation

the model on images from all $\mathscr{L}$ landmarks. This ensures that predictions from all $\mathscr{L}$ landmark images of a particular class are considered. The final prediction is determined using a majority voting fusion function, where the label appearing most frequently across the different landmarks is selected. This workflow is illustrated in Figure 9.

In the case of a tie during the voting, we applied two tie-breaking strategies that considered the model's confidence scores:

1. **Probability maximization**: Resolve ties by selecting the label with the highest probability (confidence) among the tied labels. As an example, if two samples support class A with confidence scores $p1$ and $p2$ and two samples support class B with confidence scores $p3$ and $p4$, then final prediction is the label with confidence score $p$, where $p = \max(p1, p2, p3, p4)$.
2. **Average probability maximization**: Resolve ties by selecting the label with the highest average confidence among the tied labels. For example, if two samples support class A with confidence scores $p1$ and $p2$ and two samples support class B with confidence scores $p3$ and $p4$, then final prediction is the label associated with $p$, where $p = \max(\text{mean}(p1, p2), \text{mean}(p3, p4))$.

In summary, when resolving ties in majority voting, the highest probability method selects the label with the highest individual probability, ensuring that the most confident prediction is chosen. In contrast, the highest average probability method takes into account the average probability across all tied landmarks, providing a more balanced approach by considering the overall confidence for each label.

### 4.6 Ensemble learning

Since we had 21 datasets, we also considered ensemble learning in this study. In ensemble learning, multiple models are trained on a dataset. During testing, it combines the predictions of multiple models to improve the overall performance compared to individual models. The datasets for these models can either be the same or different subsets of the original data, depending on the ensemble method employed (e.g., bagging, boosting, or stacking).

In this study, we used the datasets corresponding to the top-performing $\mathscr{L}$ landmarks to train $\mathscr{L}$ models. Note that, unlike the consolidated dataset, these $\mathscr{L}$ datasets were not combined and were used individually to train their separate model as shown in Figure 10.

Since our original video dataset contained only one sample per class, we couldn't apply traditional ensemble learning techniques. However, after the detection of hand landmarks, we could generate multiple datasets from the original data. During ensemble learning, each dataset was used independently to train a separate model, resulting in multiple trained models. For instance, we trained five CNN models, each on one of the top-performing $\mathscr{L} = 5$ landmark datasets, with each model dedicated to a single dataset. The final results were determined using majority voting, as explained earlier. This ensemble learning workflow is illustrated in Figure 10.

After achieving significant improvements with ensemble learning, we applied an additional technique to build on this success. We hypothesized that if the top five landmarks could enhance performance, extending the approach to fewer or more landmarks might produce even better results. Consequently, we implemented an exhaustive strategy where we iteratively applied ensemble learning on the top $N$ landmarks, with $N$ ranging from 2 to 21. This ensemble learning algorithm is presented in Algorithm 3 and the workflow is illustrated in Figure 11.

**Algorithm 3** Ensemble learning: Top $N$ landmarks

**for** $i \leftarrow 2$ to $N$ **do**
  $trainDir \leftarrow$ Load training dataset corresponding to the top $i^{th}$ landmark
  $model \leftarrow$ Train the classification model using $trainDir$
  $testDir \leftarrow$ Load testing dataset corresponding to the top $i^{th}$ landmark
  $results[i] \leftarrow$ Test $model$ on $testDir$
  $count \leftarrow 0$
  **for** each class $C$ in the dataset **do**
    $topLabel \leftarrow$ Get the top label via majority voting
    **if** $C == topLabel$ **then**
      $count \leftarrow count + 1$
    **end if**
  **end for**
  $accuracies[i] \leftarrow \frac{count}{numberOfClasses}$
**end for**

## 5 Experiments, results, and discussion

This section outlines the experimental setup, procedures, and results of our study. The experiments were conducted to evaluate the effectiveness of our image-based PSLR system. Our aim was to rigorously test our hypotheses and validate the proposed methodologies through a series of experiments. We detail the setup, datasets generated and utilized, experimental procedures, evaluation metrics, and statistical analysis applied.

### 5.1 Experimental setup

The experiments were conducted on a MacBook Pro (2020) equipped with a 1.4 GHz Quad-Core Intel Core i5 processor, 8 GB of 2133 MHz LPDDR3 memory, and Intel Iris Plus Graphics with 1536 MB. The software environment

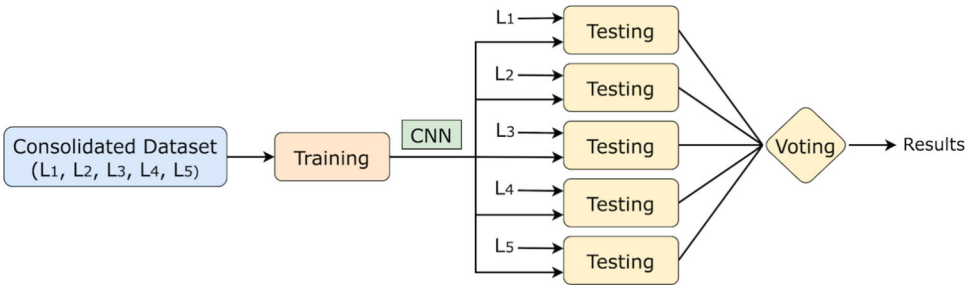



**Fig. 9** Machine learning workflow where a single recognition model is trained on the consolidated dataset generated from the top 5 landmarks (referred to as $L_1$, $L_2$, $L_3$, $L_4$, and $L_5$) and tested on the corresponding test datasets for each landmark

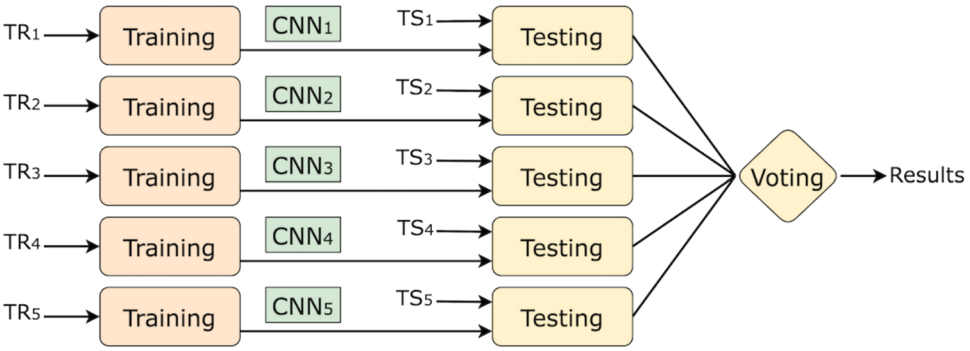



**Fig. 10** Ensemble learning workflow where 5 recognition models are trained on the 5 training sets ($TR_1$, $TR_2$, $TR_3$, $TR_4$, and $TR_5$) corresponding to the top 5 landmarks ($L_1$, $L_2$, $L_3$, $L_4$, and $L_5$, respectively). These 5 models are then tested on the test datasets corresponding to each of these landmarks

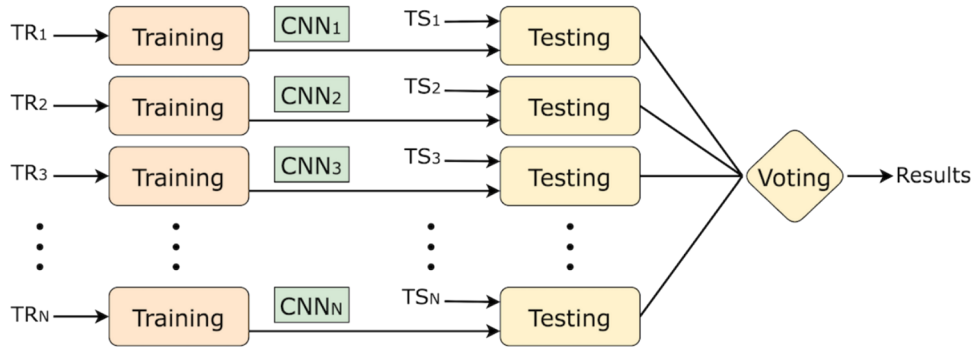



**Fig. 11** Ensemble learning applied on top $N$ landmarks where $N$ recognition models are trained on the datasets corresponding to the top $N$ landmarks (referred to as $TR_1$, $TR_2$, $TR_3$,... $TR_N$). These $N$ models are then tested on their respective test datasets corresponding to each of these landmarks (e.g., $TS_1$ for $L_1$ landmark, $TS_2$ for $L_2$ landmark, and so on)

included Python (Version 3.8.5) as the programming language. The key libraries utilized for the experiments were NumPy 1.23.1, OpenCV 4.8.1.78, TensorFlow 2.8.0, scikit-learn 0.23.2, and Mediapipe 0.9.1.0. All experiments were conducted on Visual Studio Code (version 1.90.1)

## 5.2 Initial experimentation

In the initial experimentation phase, we aimed to evaluate the performance of our classification models-CNN, SVM, and TL-on individual hand landmark datasets. Each model was first trained and tested on datasets corresponding to each specific landmark separately to assess their recognition capabilities.

For the CNN model, the experiments were repeated five times to ensure a robust evaluation. The average accuracies from these runs were reported, and the CNN model was trained for 50 epochs during each trial. In contrast to the CNN model, SVMs are deterministic algorithms, meaning they produce consistent results when trained on the same dataset with identical parameters. As a result, each SVM experiment was conducted only once, without repetition, since multiple runs were unnecessary for assessing its performance on individual landmarks. The third classification technique, TL, was also evaluated in the same manner. After setting up the initial parameters for TL, we assessed its performance across all hand landmarks, similar to the approach used for the other models.

The results of these initial experiments are summarized in Table 10. The highest accuracy achieved by the CNN model was **77%**, corresponding to the MIDDLE_FINGER_TIP (12) landmark. For the SVM model, the highest accuracy was **51.25%**, which was observed for the RING_FINGER_PIP (14) landmark. Finally, the TL model reached its highest accuracy of **36.25%**, which was recorded for both the INDEX_FINGER_DIP (7) and INDEX_FINGER_TIP (8) landmarks. The results obtained using TL were expected. Transfer learning, while successful in many image classification tasks, often relies on feature extraction components pre-trained on large-scale datasets like ImageNet, which pri-

**Table 10** Results of the initial experiments with CNN, SVM, and TL

| Train dataset | Test dataset | CNN | SVM | TL |
|---|---|---|---|---|
| ds_0_tr | ds_0_ts | 67.50 | 41.25 | 26.25 |
| ds_1_tr | ds_1_ts | 74.75 | 41.25 | 22.50 |
| ds_2_tr | ds_2_ts | 74.00 | 45.00 | 21.25 |
| ds_3_tr | ds_3_ts | 74.50 | 43.75 | 22.50 |
| ds_4_tr | ds_4_ts | 75.25 | 45.00 | 17.50 |
| ds_5_tr | ds_5_ts | 74.25 | 38.75 | 23.75 |
| ds_6_tr | ds_6_ts | 70.25 | 38.75 | 26.25 |
| ds_7_tr | ds_7_ts | 76.25 | 41.25 | **36.25** |
| ds_8_tr | ds_8_ts | 75.50 | 45.00 | **36.25** |
| ds_9_tr | ds_9_ts | 68.25 | 37.50 | 18.75 |
| ds_10_tr | ds_10_ts | 72.75 | 33.75 | 33.75 |
| ds_11_tr | ds_11_ts | 76.50 | 37.50 | 35.00 |
| ds_12_tr | ds_12_ts | **77.00** | 45.00 | 25.00 |
| ds_13_tr | ds_13_ts | 72.25 | 40.00 | 22.50 |
| ds_14_tr | ds_14_ts | 73.00 | **51.25** | 31.25 |
| ds_15_tr | ds_15_ts | 71.50 | 46.25 | 25.00 |
| ds_16_tr | ds_16_ts | 75.50 | 42.50 | 21.25 |
| ds_17_tr | ds_17_ts | 73.25 | 42.50 | 22.50 |
| ds_18_tr | ds_18_ts | 74.25 | 48.75 | 33.75 |
| ds_19_tr | ds_19_ts | 73.00 | 48.75 | 25.00 |
| ds_20_tr | ds_20_ts | 72.75 | 48.75 | 22.50 |

marily consist of natural images. These pre-trained models may not generalize well to specialized domains such as handwritten text or sign language, where the visual features differ significantly from natural images. Studies on handwritten text recognition [50, 52] also support this argument. This TL limitation is particularly pronounced when the target dataset is small, as is the case with the PSL dataset, leading to suboptimal performance in transfer learning scenarios.

## 5.3 Landmark selection

Following the initial experimentation phase, the next step was to identify the best-performing landmarks for each classification model. To do this, we ranked the landmarks based on their accuracy scores for each model (Table 11) and selected the top five for further analysis.

From Table 11, it is clear that for the CNN model, the top five landmarks are MIDDLE_FINGER_TIP (12), MIDDLE_FINGER_DIP (11), INDEX_FINGER_DIP (7), INDEX_FINGER_TIP (8), and RING_FINGER_TIP (16). This result is intuitive, as the index and middle fingers are the most commonly used, with their tips being the most visible parts. As a result, the most significant landmarks for the CNN model turn out to be the tips and distal interphalangeal joints (DIP) of the index and middle fingers, along with the tip of the ring finger.

For the SVM model, the results indicate that the landmarks of the ring and pinky fingers are the most critical for achieving high accuracy. The top five landmarks identified were RING_FINGER_PIP (14), PINKY_FINGER_PIP (18), PINKY_FINGER_DIP (19), PINKY_FINGER_TIP (20), and RING_FINGER_DIP (15).

In the case of transfer learning, the top five landmarks identified were INDEX_FINGER_DIP (7), INDEX_FINGER_TIP (8), MIDDLE_FINGER_DIP (11), MIDDLE_FINGER_PIP (10), and PINKY_FINGER_PIP (18). It is worth mentioning that the prominence of the index and middle fingers in the TL ranking aligns with the results from the CNN model.

## 5.4 Feature aggregation

The identification of the top landmarks allowed us to proceed with testing our feature aggregation strategy. First, we generated three datasets-ds_cnn_fa, ds_svm_fa, and ds_tl_fa-using our feature aggregation approach. Each dataset contained the top five landmarks identified for CNN, SVM, and TL, respectively. Each model was trained and tested on its corresponding feature aggregation training and testing datasets. The details of these experiments are presented in Table 12.

During feature aggregation, the top five landmarks were selected, resulting in an accuracy of 82.75%. This suggests that combining multiple landmarks provides a more comprehensive representation of the gesture, capturing both spatial and temporal information. The index and middle finger tips are very critical landmarks, which is why this combination performed better. The prominence of the index and middle fingers has also been emphasized in other studies such as [8, 53] for distinguishing between signs.

The results show that the CNN model demonstrated a 5.75% improvement compared to its initial performance (Table 10), achieving an accuracy of **82.75%**. In contrast, the accuracy of the SVM model decreased by 3.75%, reaching an accuracy of **47.5%**. Notably, this accuracy was lower than the top four accuracies from the initial SVM experiments (Table 10), suggesting that feature aggregation did not perform well with SVM. Similarly, feature aggregation was not effective for TL either, dropping its accuracy by 12.5% to **23.75%**. In conclusion, while the proposed feature aggregation strategy proved effective for CNN, it was less suitable for SVM and TL.

As discussed earlier, three of these top five landmarks for CNN are the tips of the index, middle, and ring fingers, which are the most visible parts during actions. Building on this observation, we hypothesized that a dataset consisting of all five finger tips might yield better results for CNN. To test this hypothesis, we generated another dataset which included all five fingertips: THUMB_TIP (4),

**Table 11** Hand landmark rankings of initial CNN, SVM and transfer learning experimentation

| Rank | CNN | SVM | Transfer Learning |
|---|---|---|---|
| 1 | MIDDLE_FINGER_TIP (12) | RING_FINGER_PIP (14) | INDEX_FINGER_DIP (7) |
| 2 | MIDDLE_FINGER_DIP (11) | PINKY_FINGER_PIP (18) | INDEX_FINGER_TIP (8) |
| 3 | INDEX_FINGER_DIP (7) | PINKY_FINGER_DIP (19) | MIDDLE_FINGER_DIP (11) |
| 4 | INDEX_FINGER_TIP (8) | PINKY_FINGER_TIP (20) | MIDDLE_FINGER_PIP (10) |
| 5 | RING_FINGER_TIP (16) | RING_FINGER_DIP (15) | PINKY_FINGER_PIP (18) |
| 6 | THUMB_TIP (4) | THUMB_MCP (2) | RING_FINGER_PIP (14) |
| 7 | THUMB_CMC (1) | THUMB_TIP (4) | WRIST (0) |
| 8 | THUMB_IP (3) | INDEX_FINGER_TIP (8) | INDEX_FINGER_PIP (6) |
| 9 | INDEX_FINGER_MCP (5) | MIDDLE_FINGER_TIP (12) | MIDDLE_FINGER_TIP (12) |
| 10 | PINKY_FINGER_PIP (18) | THUMB_IP (3) | RING_FINGER_DIP (15) |
| 11 | THUMB_MCP (2) | RING_FINGER_TIP (16) | PINKY_FINGER_DIP (19) |
| 12 | PINKY_FINGER_MCP (17) | PINKY_FINGER_MCP (17) | INDEX_FINGER_MCP (5) |
| 13 | RING_FINGER_PIP (14) | WRIST (0) | THUMB_CMC (1) |
| 14 | PINKY_FINGER_DIP (19) | THUMB_CMC (1) | THUMB_IP (3) |
| 15 | MIDDLE_FINGER_PIP (10) | INDEX_FINGER_DIP (7) | RING_FINGER_MCP (13) |
| 16 | PINKY_FINGER_TIP (20) | RING_FINGER_MCP (13) | PINKY_FINGER_MCP (17) |
| 17 | RING_FINGER_MCP (13) | INDEX_FINGER_MCP (5) | PINKY_FINGER_TIP (20) |
| 18 | RING_FINGER_DIP (15) | INDEX_FINGER_PIP (6) | THUMB_MCP (2) |
| 19 | INDEX_FINGER_PIP (6) | MIDDLE_FINGER_MCP (9) | RING_FINGER_TIP (16) |
| 20 | MIDDLE_FINGER_MCP (9) | MIDDLE_FINGER_DIP (11) | MIDDLE_FINGER_MCP (9) |
| 21 | WRIST (0) | MIDDLE_FINGER_PIP (10) | THUMB_TIP (4) |

**Table 12** Models' performance on the datasets generated using the feature aggregation approach

| Model | Top Landmark IDs | Train Dataset | Test Dataset | Accuracy | Diff. |
|---|---|---|---|---|---|
| CNN | [12, 11, 7, 8, 16] | ds_cnn_fa_tr | ds_cnn_fa_ts | 82.75 | +5.75 |
| SVM | [14, 18, 19, 20, 15] | ds_svm_fa_tr | ds_svm_fa_ts | 47.50 | -3.75 |
| TL | [7, 8, 11, 10, 18] | ds_tl_fa_tr | ds_tl_fa_ts | 23.75 | -12.50 |

INDEX_FINGER_TIP (8), MIDDLE_FINGER_TIP (12), RING_FINGER_TIP (16), and PINKY_FINGER_TIP (20). However, this experiment resulted in an accuracy of **74.75%** only.

## 5.5 Dataset consolidation

We evaluated our dataset consolidation strategy by generating three datasets: ds_cnn_dc, ds_svm_dc, and ds_tl_dc. Both training and testing sets were generated for each dataset, with each set containing five samples per class. To evaluate the consolidated datasets, we applied the testing strategy described in Section 4.5.1. Specifically, we tested each model on the corresponding test dataset, which contained five test samples, and determined the final output through majority voting.

For the CNN model, we trained it on ds_cnn_dc_tr and tested it on ds_cnn_dc_ts. This experiment resulted in a top accuracy of **81.5%**. As part of the comparative analysis, we also tested the CNN model individually on each of the landmarks. The results of all these experiments are summarized in Table 13.

**Table 13** CNN performance on various datasets when trained on consolidated dataset

| Train Dataset | Test Dataset | Accuracy |
|---|---|---|
| ds_cnn_dc_tr | ds_cnn_dc_ts | **81.50** |
| ds_cnn_dc_tr | ds_12_ts | 76.25 |
| ds_cnn_dc_tr | ds_11_ts | 78.75 |
| ds_cnn_dc_tr | ds_7_ts | 75.25 |
| ds_cnn_dc_tr | ds_8_ts | 75.75 |
| ds_cnn_dc_tr | ds_16_ts | 65.25 |

Similarly, the SVM model was evaluated on the ds_svm_dc dataset, which was generated using the dataset consolidation approach. This experiment yielded an accuracy of only **23.75%**. Based on these results, we concluded that both

**Table 14** Results of dataset consolidation with transfer learning, where ds_tl_dc_*X*_tr is the dataset generated by combining the datasets corresponding to the top *X* landmarks.

| Train dataset | Test dataset | Test Accuracy |
|---|---|---|
| ds_tl_dc_5_tr | ds_tl_dc_5_ts | 55.00 |
| ds_tl_dc_6_tr | ds_tl_dc_5_ts | 58.75 |
| ds_tl_dc_7_tr | ds_tl_dc_5_ts | 57.50 |
| ds_tl_dc_8_tr | ds_tl_dc_5_ts | 57.50 |
| ds_tl_dc_9_tr | ds_tl_dc_5_ts | 55.00 |
| ds_tl_dc_10_tr | ds_tl_dc_5_ts | 62.50 |
| ds_tl_dc_11_tr | ds_tl_dc_5_ts | 56.25 |
| ds_tl_dc_12_tr | ds_tl_dc_5_ts | **65.00** |
| ds_tl_dc_13_tr | ds_tl_dc_5_ts | 56.25 |
| ds_tl_dc_14_tr | ds_tl_dc_5_ts | 56.25 |
| ds_tl_dc_15_tr | ds_tl_dc_5_ts | 56.25 |
| ds_tl_dc_16_tr | ds_tl_dc_5_ts | 61.25 |
| ds_tl_dc_17_tr | ds_tl_dc_5_ts | 61.25 |
| ds_tl_dc_18_tr | ds_tl_dc_5_ts | 58.75 |
| ds_tl_dc_19_tr | ds_tl_dc_5_ts | 57.50 |
| ds_tl_dc_20_tr | ds_tl_dc_5_ts | 55.00 |
| ds_tl_dc_21_tr | ds_tl_dc_5_ts | 57.50 |

techniques-feature aggregation and dataset consolidation-were ineffective for SVMs, in contrast to their effectiveness for CNNs.

For transfer learning, we applied the dataset consolidation technique in an incremental manner. Rather than generating a single dataset with the top five landmarks, we created multiple datasets, with the $N^{th}$ dataset containing the top $N$ landmarks. For example, for the top five landmarks, we generated the dataset ds_tl_dc_5_tr; for the top six landmarks, we generated ds_tl_dc_6_tr, and so on. In total, 17 datasets were generated, ranging from the top five landmarks to all 21 landmarks.

The transfer learning model was trained on each of these consolidated datasets individually. All trained models were evaluated using a consistent test dataset, ds_tl_dc_5_ts. The results of this experiment, as shown in Table 14, indicate a top accuracy of **65.00%**, achieved with the dataset containing the top 12 landmarks. This represents a 28.75% improvement over the highest accuracy obtained in the initial experiments (Table 10). The relationship between the growth in training time for the transfer learning technique and the number of landmarks in the consolidated datasets is illustrated in Figure 12.

## 5.6 Ensemble learning

To employ ensemble learning, we aimed to train multiple models using different datasets. Since we had already generated several datasets based on hand landmarks, we utilized these datasets to train multiple instances of our models.

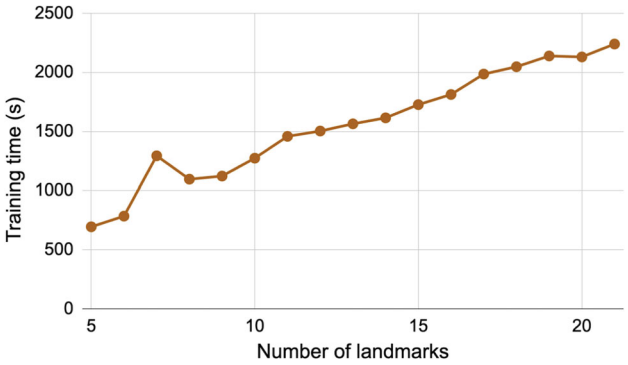



**Fig. 12** Growth in training time for transfer learning as the number of landmarks increases in the consolidated datasets

**Table 15** Models' performance using ensemble learning with top *N* landmarks

| N | Description | CNN | SVM | TL |
|---|---|---|---|---|
| 2 | Top 2 landmarks | 77.50 | 45.00 | 41.25 |
| 3 | Top 3 landmarks | 77.50 | 53.75 | 42.50 |
| 4 | Top 4 landmarks | 81.25 | 55.00 | 50.00 |
| 5 | Top 5 landmarks | 83.75 | 60.00 | 52.50 |
| 6 | Top 6 landmarks | 83.75 | 60.00 | 55.00 |
| 7 | Top 7 landmarks | 88.75 | 61.25 | 55.00 |
| 8 | Top 8 landmarks | **92.50** | 62.50 | 57.50 |
| 9 | Top 9 landmarks | 90.00 | 62.50 | 57.50 |
| 10 | Top 10 landmarks | 88.75 | 65.00 | 56.25 |
| 11 | Top 11 landmarks | 88.75 | 66.25 | 57.50 |
| 12 | Top 12 landmarks | 87.50 | 65.00 | 57.50 |
| 13 | Top 13 landmarks | 91.25 | 71.25 | 57.50 |
| 14 | Top 14 landmarks | 88.75 | 71.25 | 57.50 |
| 15 | Top 15 landmarks | 86.25 | 71.25 | 61.25 |
| 16 | Top 16 landmarks | 87.50 | 71.25 | 60.00 |
| 17 | Top 17 landmarks | 86.25 | **72.50** | 60.00 |
| 18 | Top 18 landmarks | 86.25 | 71.25 | 60.00 |
| 19 | Top 19 landmarks | 87.50 | 70.00 | 61.25 |
| 20 | Top 20 landmarks | 87.50 | 70.00 | **62.50** |
| 21 | Top 21 landmarks | 85.00 | 70.00 | **62.50** |

First, we applied ensemble learning to the CNN model, training five instances of CNN on the top five landmarks identified in Table 11. Each of these five datasets (ds_7_tr, ds_8_tr, ds_11_tr, ds_12_tr, and ds_16_tr) was used to train a distinct CNN model. The models were then evaluated on their corresponding test datasets (ds_7_ts, ds_8_ts, ds_11_ts, ds_12_ts, and ds_16_ts), with the final output determined by majority voting. This approach resulted in a maximum accuracy of **83.75%**.

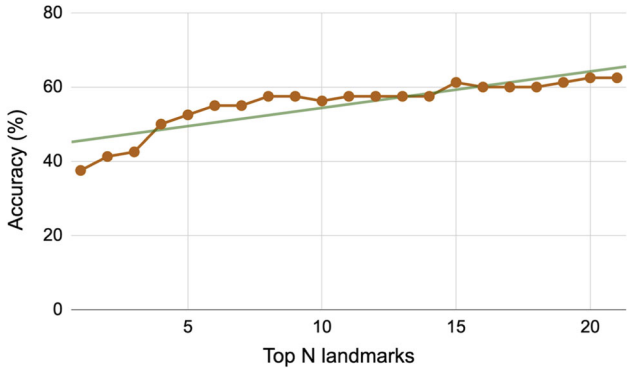



**Fig. 13** Accuracy improvement as the number of top *N* landmarks increases

Encouraged by this significant improvement, we expanded the ensemble learning strategy to include ten CNN models, each trained on one of the top ten landmarks individually. This experiment led to a further improvement in accuracy, reaching **88.75%**. The results of all these experiments are summarized in Table 15. As discussed in Section 4.6, we employed an exhaustive strategy to explore whether increasing the number of CNNs would yield even higher accuracy or if a similar accuracy could be achieved with fewer models. We iteratively applied ensemble learning on the top *N* landmarks, where N ranged from 2 to 21. As presented in Table 15, the highest accuracy was obtained using the top 8 landmarks, achieving **92.50%**. This marks a 15.5% improvement over the best accuracy of 77% obtained with the original datasets (Table 10). Additionally, for further statistical analysis of the best results obtained with CNN using the top 8 landmarks, we have incorporated other performance metrics, including precision, recall, and F1 score for all classes, as detailed in Table 16. The overall F1 score of **90%** and a confidence interval of (86.73%, 98.27%) further validates the effectiveness of our approach.

Similarly, we evaluated ensemble learning with SVM, focusing on the top *N* landmarks, where *N* ranged from 2 to 21. As shown in Table 15, the accuracy steadily increased, reaching its peak at the top 17 landmarks. Ultimately, we achieved an accuracy of **72.5%**, representing a significant improvement of 21.25% over the highest accuracy of 51.25% obtained during the initial experiments (Table 10).

Ensemble learning also proved to be highly effective for transfer learning. In contrast to CNN and SVM, the performance of TL continued to improve as the number of top landmarks increased, ultimately achieving the best accuracy with all landmarks. The final accuracy reached **62.50%**, marking an improvement of 26.25% compared to the initial results (Table 10). The trend of increasing accuracy with the increasing number of top *N* landmarks is illustrated in Figure 13.

## 5.7 Comparison with previous studies that used the PSL dataset

The only study that used the PSL dataset was [1] that employed three action recognition models to develop a PSL recognition system. This approach required a substantial amount of storage space to store the video data and significant computational power to train and test the action recognition models. In this study, we converted video gesture data into images using hand landmarks.

We provide a comparison with [1] using a number of parameters that include the number of samples, storage space (in MBs), average training time (in seconds), and the highest accuracy achieved by each model.

Table 17 provides a comparison between the six models used in the two studies. The training dataset consisted of the original 80 samples. Since all action recognition models shared the same dataset and similarly, all image recognition models used the same dataset, their required storage spaces are consistent: 13.8 MBs for action recognition models and 0.34 MBs for image recognition models.

Subsequently, we present a comparison of the optimal strategies proposed for these six models, along with their corresponding time and space requirements in Table 18. The study by Hamza et al. [1] achieved optimal results using different combinations of data augmentation techniques. Whereas for the proposed system the best results were obtained using different ensemble techniques. The optimal strategies that yielded the best results for both studies are summarized below:

- **C3D:** Trained on an augmented dataset combining Original (O), Brightness (B), Noise (N), Rotation (R), Scaling (S), and Translation (T) transformations.
- **I3D:** Trained on an augmented dataset combining Original (O), Rotation (R), and Translation (T) transformations.
- **TSM:** Trained on an augmented dataset combining Original (O), Brightness (B), Noise (N), Rotation (R), Scaling (S), and Translation (T) transformations.
- **CNN:** Utilized ensemble learning with the top 8 hand landmarks.
- **SVM:** Utilized ensemble learning with the top 17 hand landmarks.
- **Transfer learning:** Employed dataset consolidation with the top 12 hand landmarks.

## 5.8 Comparative analysis of resource efficiency and performance

It can be observed that the storage requirements and training times for action recognition models are significantly greater than those for image recognition models. This is intuitively

**Table 16** Precision, recall, and F1 score for all classes with CNN using ensemble learning with top 8 landmarks

| Class | Precision | Recall | F1 Score | Class | Precision | Recall | F1 Score |
|---|---|---|---|---|---|---|---|
| 10 | 0.50 | 1.00 | 0.67 | Near | 1.00 | 1.00 | 1.00 |
| 50 | 0.00 | 0.00 | 0.00 | Off | 0.00 | 0.00 | 0.00 |
| 100 | 1.00 | 1.00 | 1.00 | Often | 1.00 | 1.00 | 1.00 |
| Abnormal | 1.00 | 1.00 | 1.00 | One | 1.00 | 1.00 | 1.00 |
| Absolutely | 1.00 | 1.00 | 1.00 | Our | 1.00 | 1.00 | 1.00 |
| Afraid | 1.00 | 1.00 | 1.00 | Outdoors | 1.00 | 1.00 | 1.00 |
| Almost | 1.00 | 1.00 | 1.00 | Outside | 1.00 | 1.00 | 1.00 |
| Ancient | 1.00 | 1.00 | 1.00 | Parents | 1.00 | 1.00 | 1.00 |
| Annual | 1.00 | 1.00 | 1.00 | Request | 1.00 | 1.00 | 1.00 |
| Another | 1.00 | 1.00 | 1.00 | Roundabout | 1.00 | 1.00 | 1.00 |
| Any | 1.00 | 1.00 | 1.00 | Say | 0.00 | 0.00 | 0.00 |
| Because | 1.00 | 1.00 | 1.00 | She | 1.00 | 1.00 | 1.00 |
| Both | 1.00 | 1.00 | 1.00 | Sister | 1.00 | 1.00 | 1.00 |
| Brain | 1.00 | 1.00 | 1.00 | So (Accentuator) | 1.00 | 1.00 | 1.00 |
| Children | 1.00 | 1.00 | 1.00 | So (In Order To) | 1.00 | 1.00 | 1.00 |
| Come | 1.00 | 1.00 | 1.00 | Some | 1.00 | 1.00 | 1.00 |
| Continuously | 0.50 | 1.00 | 0.67 | Soon | 1.00 | 1.00 | 1.00 |
| Do | 1.00 | 1.00 | 1.00 | Subsequent | 1.00 | 1.00 | 1.00 |
| Dry | 1.00 | 1.00 | 1.00 | Subway | 1.00 | 1.00 | 1.00 |
| Elevator | 0.00 | 0.00 | 0.00 | Sufficient | 1.00 | 1.00 | 1.00 |
| Empty | 1.00 | 1.00 | 1.00 | There | 1.00 | 1.00 | 1.00 |
| Eye | 1.00 | 1.00 | 1.00 | This | 0.50 | 1.00 | 0.67 |
| Father | 1.00 | 1.00 | 1.00 | Thoughtful | 0.50 | 1.00 | 0.67 |
| Few | 0.50 | 1.00 | 0.67 | Tongue | 1.00 | 1.00 | 1.00 |
| From | 1.00 | 1.00 | 1.00 | Trust | 1.00 | 1.00 | 1.00 |
| Go | 1.00 | 1.00 | 1.00 | Truthful | 1.00 | 1.00 | 1.00 |
| He | 1.00 | 1.00 | 1.00 | Universal | 1.00 | 1.00 | 1.00 |
| Hear | 1.00 | 1.00 | 1.00 | Up | 0.00 | 0.00 | 0.00 |
| Heart | 1.00 | 1.00 | 1.00 | Upward | 1.00 | 1.00 | 1.00 |
| However | 1.00 | 1.00 | 1.00 | Usually | 1.00 | 1.00 | 1.00 |
| I | 1.00 | 1.00 | 1.00 | Walk | 1.00 | 1.00 | 1.00 |
| Jeep | 1.00 | 1.00 | 1.00 | Warm | 1.00 | 1.00 | 1.00 |
| Knock | 1.00 | 1.00 | 1.00 | We | 1.00 | 1.00 | 1.00 |
| Lakh | 1.00 | 1.00 | 1.00 | Weak | 1.00 | 1.00 | 1.00 |
| Literally | 1.00 | 1.00 | 1.00 | Without | 0.00 | 0.00 | 0.00 |
| Mehr | 1.00 | 1.00 | 1.00 | Woman | 1.00 | 1.00 | 1.00 |
| Mine | 1.00 | 1.00 | 1.00 | Work | 1.00 | 1.00 | 1.00 |
| Mother | 1.00 | 1.00 | 1.00 | Worthy | 1.00 | 1.00 | 1.00 |
| Mouth | 0.50 | 1.00 | 0.67 | Yellow Light | 1.00 | 1.00 | 1.00 |
| Move | 1.00 | 1.00 | 1.00 | You | 1.00 | 1.00 | 1.00 |

expected due to the more complex architectures of action recognition models. Specifically, if we compare the best results from both approaches - C3D achieving 94.58% accuracy and CNN achieving 92.5% accuracy - we see that the former was trained on a dataset requiring 593.1 MB of storage, while the latter was trained on a dataset of only 2.8 MB.

In terms of training time, the C3D model took 1702 seconds to train, whereas the CNN model completed training in just 58.08 seconds. Thus, it is evident that the image recognition models achieved comparable accuracy while requiring substantially fewer resources. This highlights the efficiency of image recognition-based methods for PSL recognition,

**Table 17** Comparison of base experiments for action and image recognition models

| Recognition | Model | Samples | Storage (MBs) | Training Time (s) | Accuracy |
|---|---|---|---|---|---|
| Action recognition [1] | C3D | 80 | 13.8 | 546.50 | 70.00 |
| | I3D | 80 | 13.8 | 902.00 | 81.25 |
| | TSM | 80 | 13.8 | 699.00 | 40.00 |
| Image recognition | CNN | 80 | 0.34 | 7.97 | 82.75 |
| | SVM | 80 | 0.34 | 2.70 | 51.25 |
| | TL | 80 | 0.34 | 224.10 | 36.25 |

**Table 18** Comparison of optimal strategies for action and image recognition models

| Recognition | Model | Samples | Storage (MBs) | Training Time (s) | Accuracy |
|---|---|---|---|---|---|
| Action recognition [1] | C3D | 480 | 593.1 | 1702 | 94.58 |
| | I3D | 240 | 81.2 | 2006 | 87.50 |
| | TSM | 480 | 593.1 | 2830 | 57.08 |
| Image recognition | CNN | 640 | 2.8 | 58 | 92.50 |
| | SVM | 1360 | 5.9 | 45 | 72.50 |
| | TL | 960 | 4.2 | 1502 | 65.00 |

particularly when computational and storage resources are limited.

This analysis highlights the real-time feasibility of our proposed system by evaluating its computational efficiency and deployment potential on edge devices. Our transformation from an action recognition task (which requires high storage and computational power) to an image recognition-based approach significantly reduces computational demands while maintaining comparable performance. This makes our approach well-suited for real-time applications. Moreover, our methodologies can be deployed on resource-constrained devices such as smartphones, tablets, and microprocessors.

Since CNN inference on images is considerably faster than processing continuous video frames, our approach ensures low latency and efficient execution. Furthermore, the optimization strategies we proposed to achieve outstanding results do not require extensive computational resources. Additionally, our framework can be hosted as a cloud-based API, enabling lightweight deployment on edge devices without requiring substantial onboard processing power. Based on these factors, we conclude that our method is highly suitable for real-time PSL recognition.

## 6 Conclusion and Future Work

PSL is the primary language for around 250,000 deaf individuals in Pakistan, developed with its own unique grammar, syntax, and vocabulary. Despite its significance, PSL has not received the same level of attention or development as other sign languages like ASL or CSL. While the PSL Dictionary dataset is publicly available, no significant progress has been made to create an efficient PSLR system. Existing efforts are limited to static image recognition, focusing mainly on the alphabet and digits. There is a vital need for a comprehensive PSLR system to bridge the communication gap between the deaf and hearing populations. However, the PSL Dictionary dataset is very limited providing only two samples per sign, which poses a significant challenge in building a robust and reliable system.

The PSL Dictionary dataset consists of videos that require substantial storage capacity. Additionally, training and testing action recognition models on video datasets require significant computational resources, typically necessitating a GPU. To address these challenges, we proposed a novel approach that eliminates the dependency on video datasets. This strategy involves converting each video into a single image and leveraging image recognition models to develop the PSL recognition system. Specifically, we detected hand landmarks in the video frames and mapped their movements onto a binary image, creating a compact representation of the gesture.

Using the generated image dataset, we developed a lightweight yet efficient PSLR system based on image recognition. We utilized three machine learning techniques: CNN, SVM, and transfer learning. To further enhance the system's accuracy, we applied various aggregation and ensembling techniques at different stages of the machine learning process, which proved highly effective for the proposed PSLR system. Additionally, we compared the two PSLR systems developed in this work: one based on action recognition and the other on image recognition. Our analysis concluded that the image recognition models achieved comparable accuracy while requiring significantly fewer computational and storage resources.

In this study, we used the dataset consisting of only one-handed signs due to MediaPipe's inability to differentiate between the left and right hands. We specifically needed the hand information for missing-point algorithms discussed in Section 3.12 (Discontinuous signs). Fortunately, converting discontinuous sign images to a continuous form by adding missing points did not improve the accuracy. For future studies, both handed signs can be used. Also. for future work, the size of the dataset can be expanded from 80 samples covered in this study.

**Funding** This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

**Data Availability** The entire code along with the publicly available dataset are available at https://github.com/hmhamza/psl-dataset.

## Declarations

**Compliance with Ethical Standards** This statement is to certify that the author list is correct. The Authors also confirm that this research has not been published previously and that it is not under consideration for publication elsewhere. On behalf of all Co-Authors, the Corresponding Author shall bear full responsibility for the submission. There is no conflict of interest. This research did not involve any human participants and/or animals.

**Conflict of Interest** The authors declare that they have no relevant financial or non-financial interests to disclose. There is no personal relationship that could influence the work reported in this paper. No funding was received for conducting this study. The authors have no conflicts of interest to declare that are relevant to the content of this article.

## References

1. Hamza, H.M., Wali, A.: Pakistan sign language recognition: leveraging deep learning models with limited dataset. Machine Vision and Applications **34**(5), 71 (2023). https://doi.org/10.1007/s00138-023-01429-8
2. Koller, O., Ney, H., Bowden, R.: Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 3793–3802 (2016). https://doi.org/10.1109/CVPR.2016.412
3. Huang, J., Zhou, W., Li, H.: Video-based sign language recognition without temporal segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence* 2774–2781 (2018). https://doi.org/10.1609/aaai.v32i1.11550
4. Sukriti, et al.: Deep learning for indian sign language recognition: A systematic review. Pattern Recognition Letters **140**, 72–80 (2020). https://doi.org/10.1016/j.patrec.2020.08.010
5. Wali, A., Naseer, A., Tamoor, M., Gilani, S.: Recent progress in digital image restoration techniques: A review. *Digital Signal Processing* 104187 (2023)
6. Ahmed, M. A., Zaidan, B. B., Zaidan, A. A., Salih, M. M., Lakulu, M. M. b.: A review on systems-based sensory gloves for sign language recognition state of the art between 2007 and 2017. *Sensors* **18** (7), 2208 (2018)
7. Shin, J., Matsuoka, A., Hasan, M. A. M., Srizon, A. Y.: American sign language alphabet recognition by extracting feature from hand pose estimation. *Sensors* **21** (17) (2021). https://doi.org/10.3390/s21175856
8. Rastgoo, R., Kiani, K., Escalera, S.: Hand gesture recognition using deep learning: A review. Expert Systems with Applications **150**, 113336 (2020). https://doi.org/10.1016/j.eswa.2020.113336
9. Alam, M.S., Tanvir, M., Saha, D.K., Das, S.K.: Two dimensional convolutional neural network approach for real-time bangla sign language characters recognition and translation. SN Computer Science **2**(5), 387 (2021). https://doi.org/10.1007/s42979-021-00783-6
10. Ma, Y., Xu, T., Kim, K.: Two-stream mixed convolutional neural network for american sign language recognition. *Sensors* **22** (16) (2022). https://doi.org/10.3390/s22165959
11. Tamiru, N.K., Tekeba, M., Salau, A.O.: Recognition of amharic sign language with amharic alphabet signs using ann and svm. The Visual Computer **38**(5), 1703–1718 (2022). https://doi.org/10.1007/s00371-021-02099-1
12. Novianty, A., Azmi, F.: Sign language recognition using principal component analysis and support vector machine. *IJAIT (International Journal of Applied Information Technology)* **4**, 49 (2021). https://doi.org/10.25124/ijait.v4i01.3015
13. Zhang, F., *et al.*: Mediapipe hands: On-device real-time hand tracking. *CoRR* **abs/2006.10214** (2020). arXiv: 2006.10214
14. Tabish, M., Tanooli, Z.-u.-R., Shaheen, M.: Activity recognition framework in sports videos. *Multimedia Tools and Applications* 1–23 (2024)
15. Faseeh, M., Bibi, M., Khan, M. A., Kim, D.-H.: Real-time long-range object tracking based on ensembled model. *IEEE Access* (2024)
16. addin I. Sidig, A., Luqman, H., Mahmoud, S. A.: Transform-based arabic sign language recognition. *Procedia Computer Science* **117**, 2–9 (2017). https://doi.org/10.1016/j.procs.2017.10.087, arabic Computational Linguistics
17. Podder, K. K., *et al.*: Signer-independent arabic sign language recognition system using deep learning model. *Sensors* **23** (16) (2023). https://doi.org/10.3390/s23167156
18. Aldhahri, E., et al.: Arabic sign language recognition using convolutional neural network and mobilenet. Arabian Journal for Science and Engineering **48**(2), 2147–2154 (2023). https://doi.org/10.1007/s13369-022-07144-2
19. Dabwan, B. A., Jadhav, M. E., Ali, Y. A., Olayah, F. A.: *2023 International Conference on IT Innovation and Knowledge Discovery (ITIKD)*, Ch. Arabic Sign Language Recognition Using EfficientnetB1 and Transfer Learning Technique, 1–5 (IEEE, 2023)
20. Suharjito, Thiracitta, N., Gunawan, H.: Sibi sign language recognition using convolutional neural network combined with transfer learning and non-trainable parameters. *Procedia Computer Science* **179**, 72–80 (2021). https://doi.org/10.1016/j.procs.2020.12.011, 5th International Conference on Computer Science and Computational Intelligence 2020
21. Kakizaki, M., Miah, A. S. M., Hirooka, K., Shin, J.: Dynamic japanese sign language recognition throw hand pose estimation using effective feature extraction and classification approach. *Sensors* **24** (3) (2024). https://doi.org/10.3390/s24030826
22. Shin, J., *et al.*: Korean sign language recognition using transformer-based deep neural network. *Applied Sciences* **13** (5) (2023). https://doi.org/10.3390/app13053029
23. Shin, H., Kim, W. J., Jang, K.-a.: *Proceedings of the 2nd International Conference on Image and Graphics Processing*, Ch. Korean sign language recognition based on image and convolution neural network, 52-55. ICIGP '19 (Association for Computing Machinery, 2019)
24. Wadhawan, A., Kumar, P.: Deep learning-based sign language recognition system for static signs. Neural Computing and Applica-

tions **32**(12), 7957–7968 (2020). https://doi.org/10.1007/s00521-019-04691-y
25. Katoch, S., Singh, V., Tiwary, U.S.: Indian sign language recognition system using surf with svm and cnn. Array **14**, 100141 (2022). https://doi.org/10.1016/j.array.2022.100141
26. Hossen, M., Govindaiah, A., Sultana, S., Bhuiyan, A.: *2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, Ch. Bengali Sign Language Recognition Using Deep Convolutional Neural Network, 369–373 (IEEE, 2018)
27. Hasan, M. M., Srizon, A. Y., Hasan, M. A. M.: *2020 IEEE Region 10 Symposium (TENSYMP)*, Ch. Classification of Bengali Sign Language Characters by Applying a Novel Deep Convolutional Neural Network, 1303–1306 (IEEE, 2020)
28. Rahaman, M.A., et al.: Computer vision-based six layered convneural network to recognize sign language for both numeral and alphabet signs. Biomimetic Intelligence and Robotics **4**(1), 100141 (2024). https://doi.org/10.1016/j.birob.2023.100141
29. Project, P. S. L.: Psl dictionary. https://psl.org.pk/dictionary (2020). Accessed: 2022-06-08
30. Khan, N., *et al.*: A vision based approach for pakistan sign language alphabets recognition. *La Pensée* **76** (2014)
31. Raees, M., Ullah, S., Rahman, S.U., Rabbi, I.: Image based recognition of pakistan sign language. Journal of Engineering Research **4**(1), 2 (2016). https://doi.org/10.7603/s40632-016-0002-6
32. Malik, M. S. A., *et al.*: Pakistan sign language detection using pca and knn. *International Journal of Advanced Computer Science and Applications* **9** (2018)
33. Shah, F., et al.: Sign language recognition using multiple kernel learning: A case study of pakistan sign language. IEEE Access **9**, 67548–67558 (2021). https://doi.org/10.1109/ACCESS.2021.3077386
34. Mirza, M.S., Munaf, S.M., Azim, F., Ali, S., Khan, S.J.: Vision-based pakistani sign language recognition using bag-of-words and support vector machines. Scientific Reports **12**(1), 21325 (2022). https://doi.org/10.1038/s41598-022-15864-6
35. Khan, M., Naseer, A., Wali, A., Tamoor, M.: A roman urdu corpus for sentiment analysis. *The Computer Journal* bxae052 (2024)
36. Imran, A., et al.: Dataset of pakistan sign language and automatic recognition of hand configuration of urdu alphabet through machine learning. Data in Brief **36**, 107021 (2021). https://doi.org/10.1016/j.dib.2021.107021
37. Naseem, M., Sarafraz, S., Abbas, A., Haider, A.: Developing a prototype to translate pakistan sign language into text and speech while using convolutional neural networking. Journal of Education and Practice **10**(15), 10–7176 (2019)
38. Arooj, S., Altaf, S., Ahmad, S., Mahmoud, H., Mohamed, A.S.N.: Enhancing sign language recognition using cnn and sift: A case study on pakistan sign language. Journal of King Saud University - Computer and Information Sciences **36**(2), 101934 (2024). https://doi.org/10.1016/j.jksuci.2024.101934
39. Miah, A.S.M., Hasan, M.A.M., Okuyama, Y., Tomioka, Y., Shin, J.: Spatial-temporal attention with graph and general neural network-based sign language recognition. Pattern Analysis and Applications **27**(2), 37 (2024). https://doi.org/10.1007/s10044-024-01229-4
40. Shah, S.M.S., Khan, J.I., Abbas, S.H., Ghani, A.: Symmetric mean binary pattern-based pakistan sign language recognition using multiclass support vector machines. Neural Computing and Applications **35**(1), 949–972 (2023). https://doi.org/10.1007/s00521-022-07804-2
41. Jiang, Z., Zaheer, W., Wali, A., Gilani, S.: Visual sentiment analysis using data-augmented deep transfer learning techniques. *Multimedia Tools and Applications* 1–17 (2023)
42. Wali, A., Ahmad, M., Naseer, A., Tamoor, M., Gilani, S.: Stynmedgan: Medical images augmentation using a new gan model for improved diagnosis of diseases. *Journal of Intelligent & Fuzzy Systems* (Preprint), 1–18 (2023)
43. Xu, Y., Wali, A.: Handwritten pattern recognition using birds-flocking inspired data augmentation technique. *IEEE Access* (2023)
44. Cheng, K., Zhang, Y., He, X., Chen, W.: Data augmentation for skeleton-based action recognition. IEEE Transactions on Neural Networks and Learning Systems **32**(6), 2457–2470 (2020). https://doi.org/10.1109/TNNLS.2020.2999398
45. Liu, M., Liu, H., Chen, C.: Augmentation techniques for skeleton-based action recognition: A comparative study. Pattern Recognition Letters **143**, 1–8 (2021). https://doi.org/10.1016/j.patrec.2021.01.008
46. LaValle, S. M.: Rapidly-exploring random trees : a new tool for path planning. *The annual research report* (1998). https://api.semanticscholar.org/CorpusID:14744621
47. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010). https://doi.org/10.1109/TKDE.2009.191
48. Hong, H., Zaheer, W., Wali, A.: Visual sentiment analysis using data-augmented deep transfer learning techniques. Multimedia Systems **30**(2), 1–16 (2024)
49. Shahid, R., Wali, A., Bashir, M.: Next word prediction for urdu language using deep learning models. Computer Speech & Language **87**, 101635 (2024)
50. Bansal, M., Kumar, M., Sachdeva, M., Mittal, A.: Transfer learning for image classification using vgg19: Caltech-101 image data set. Journal of Ambient Intelligence and Humanized Computing **14**(4), 3609–3620 (2023). https://doi.org/10.1007/s12652-021-03488-z
51. Wali, A., Shariq, R., Shoaib, S., Amir, S., Farhan, A.A.: Recent progress in sign language recognition: a review. Machine Vision and Applications **34**(6), 127 (2023)
52. Sharma, A., Mittal, A., Singh, S.: Handwritten character recognition using transfer learning with deep convolutional neural networks. International Journal of Computer Applications **180**(4), 10–16 (2018). https://doi.org/10.5120/ijca2018916417
53. Shin, J., Matsuoka, A., Hasan, M.A.M., Srizon, A.Y.: American sign language alphabet recognition using deep learning. Sensors **21**(17), 5856 (2021). https://doi.org/10.3390/s21175856