Computer Modeling in
Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.046334

ARTICLE

# Japanese Sign Language Recognition by Combining Joint Skeleton-Based Handcrafted and Pixel-Based Deep Learning Features with Machine Learning Classification

**Jungpil Shin[1,*], Md. Al Mehedi Hasan[2], Abu Saleh Musa Miah[1], Kota Suzuki[1] and Koki Hirooka[1]**

[1]School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu, 965-8580, Japan

[2]Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi, 6204, Bangladesh

*Corresponding Author: Jungpil Shin. Email: jpshin@u-aizu.ac.jp

Received: 27 September 2023  Accepted: 22 November 2023  Published: 11 March 2024

**ABSTRACT**

Sign language recognition is vital for enhancing communication accessibility among the Deaf and hard-of-hearing communities. In Japan, approximately 360,000 individuals with hearing and speech disabilities rely on Japanese Sign Language (JSL) for communication. However, existing JSL recognition systems have faced significant performance limitations due to inherent complexities. In response to these challenges, we present a novel JSL recognition system that employs a strategic fusion approach, combining joint skeleton-based handcrafted features and pixel-based deep learning features. Our system incorporates two distinct streams: the first stream extracts crucial handcrafted features, emphasizing the capture of hand and body movements within JSL gestures. Simultaneously, a deep learning-based transfer learning stream captures hierarchical representations of JSL gestures in the second stream. Then, we concatenated the critical information of the first stream and the hierarchy of the second stream features to produce the multiple levels of the fusion features, aiming to create a comprehensive representation of the JSL gestures. After reducing the dimensionality of the feature, a feature selection approach and a kernel-based support vector machine (SVM) were used for the classification. To assess the effectiveness of our approach, we conducted extensive experiments on our Lab JSL dataset and a publicly available Arabic sign language (ArSL) dataset. Our results unequivocally demonstrate that our fusion approach significantly enhances JSL recognition accuracy and robustness compared to individual feature sets or traditional recognition methods.

**KEYWORDS**

Japanese Sign Language (JSL); hand gesture recognition; geometric feature; distance feature; angle feature; GoogleNet

## 1 Introduction

Japanese Sign Language (JSL) plays a pivotal role in enabling communication for the Deaf and hard-of-hearing communities in Japan [1–3]. Approximately 340,000 individuals rely on this visual-based language for their daily interactions [4]. Among them, 80,000 deaf communities use JSL to establish their communication to express their daily basic needs, thoughts, expressions, and requirements [5–7]. However, learning JSL presents considerable challenges, both for the Deaf community and for non-deaf individuals, primarily due to its complexity. This complexity hampers effective communication between these two communities, often necessitating human sign language translators, which are both scarce and expensive [3,8]. Many people have used a human language translator to establish communication between the deaf and non-deaf communities. However, it is difficult to find human sign language translators costly. In response to these challenges, automatic JSL recognition systems have emerged as a potential solution [9]. These systems aim to accurately interpret and translate JSL signs into meaningful communication [10]. Furthermore, in the context of the COVID-19 pandemic, the demand for touchless communication interfaces has grown, underlining the urgency of developing reliable JSL recognition technologies. Therefore, there is a growing demand for non-contact input interfaces that allow users to input data without touching their hands. JSL recognition aims to develop a feature extraction technique and classification approach to recognize the demonstrated signs and generate equivalent meanings accurately [10]. It is essential to extract various features of sign language, such as hand orientation, hand shape, movement expression, location, etc. Besides the significant process in JSL recognition for the last decades, it still faces problems because the hand gesture contains a high degree of freedom, specifically in the hand shape [9,11]. Moreover, individual distinctions in finger spelling expression have a considerable influence. The study used various feature extraction techniques to extract the mentioned information from the hand gestures. This system needed to record the actual information of the sign or dataset. Past efforts have employed feature extraction techniques and classification methods, relying on sensor-based and vision-based systems. However, sensor-based systems suffer from portability and cost limitations. In contrast, vision-based systems offer a more affordable and portable alternative, but they require effective feature extraction and classification algorithms to operate successfully. To date, researchers have explored model-based, template-based, and machine-learning approaches for hand gesture recognition. While some have employed pixel-based datasets, these approaches often grapple with challenges related to background interference, computational complexity, occlusions, and lighting conditions. Recently, skeleton-based approaches using tools like MediaPipe and OpenPose have gained attention for their potential in JSL recognition. However, achieving satisfactory performance remains an ongoing challenge, with room for improvement. We did not find any research that combined the hand-crafted feature and deep learning for JSL recognition. In this study, we propose an innovative approach to JSL recognition that capitalizes on the strengths of joint skeleton-based handcrafted features and pixel-based Convolutional Neural Network (CNN) features. The proposed method aims to enhance the accuracy and robustness of JSL recognition systems by leveraging the complementary nature of these two feature streams, which represents a significant advancement in the field of JSL recognition. We introduce an innovative fusion approach that enhances both accuracy and robustness, contributing to the state-of-the-art in sign language recognition. Our contributions extend beyond mere methodology:

- **Novelty:** Our research introduces a new and unique JSL alphabet dataset, effectively addressing the scarcity of resources in the field. This dataset is thoughtfully designed with diverse backgrounds and environmental conditions taken into account during data collection. Not only does it fill a critical gap, but it also paves the way for more comprehensive and inclusive studies in the domain of sign language recognition.
- **Methodological Innovation:** Our research introduces an innovative fusion approach that combines joint skeleton-based handcrafted features with pixel-based deep learning-based features, presenting a novel solution for JSL recognition. This fusion method not only significantly

enhances recognition performance but also showcases the potential of integrating diverse feature types in similar recognition tasks.

- In our methodology, we employed two distinct feature streams. The first stream extracted skeleton-based distance and angle features from joint coordinates, effectively capturing intrinsic joint relationships. Simultaneously, the second stream utilized GoogleNet transfer learning to extract pixel-based features from frames, enabling hierarchical representations of sign language gestures. Subsequently, we fused these two feature streams to strike a balance between interpretability and the discriminative power of deep learning, thereby enhancing the overall effectiveness of our approach. To further optimize our feature set, we implemented a feature selection algorithm, selectively retaining the most effective features while discarding irrelevant ones. Finally, we applied these reduced features as input to an support vector machine (SVM) machine-learning model equipped with multiple kernels for comprehensive evaluation. This combined approach not only represents a methodological innovation but also yields substantial improvements in JSL recognition performance.
- **Empirical Validation:** Through extensive experimentation on both our new JSL dataset and a publicly available Arabic sign language (ArSL) dataset, we establish the effectiveness of our approach. Our results unequivocally demonstrate substantial enhancements in recognition performance compared to conventional methods or individual feature sets. Our work not only showcases the effectiveness of our proposed methodology but also contributes valuable insights to the broader field of sign language recognition research.

In summary, our research contributes to the advancement of JSL recognition by presenting an innovative fusion approach that improves accuracy and robustness. Our results, based on our new JSL dataset and a publicly available Arabic sign language (ArSL) dataset, demonstrate substantial enhancements in recognition performance compared to traditional methods or individual feature sets.

## 2 Related Work

Many researchers have applied feature extraction approaches, machine learning and deep learning models in various sections, such as electromyogram (EMG) electroencephalogram (EEG) classification [12–15], hand gesture recognition and many other sign language recognition [16–19]. Most JSL recognition research has been done with image- and skeleton-based datasets. In addition, some research on Japanese Sign Language has been conducted in various forms, including specialized devices such as Leap Motion [8], and Data Glove [20,21], and research using RGB cameras. MediaPipe and OpenPose software systems have recently been used to extract the skeleton point from the RGB images. Kobayashi et al. proposed a JSL recognition system by extracting the geometric formula-based feature, specifically angle-based feature information, calculated by the cartesian coordinate joint of the skeleton point. Finally, they achieved 63.60% accuracy SVM [7]. Hossoe et al. developed a JSL recognition system to create a JSL dataset by recording 5000 samples [22]. They first applied an image generative model for increasing the synthetic training data; after that, they used CNN directly on the image pixel information and reported 93.00% performance compared to the previous studies. In the same way, Funasaka et al. collected the JSL dataset using a leap motion controller for collecting the JSL dataset, then employed a genetic algorithm and a decision tree algorithm for the feature extraction and classification and reported 74.04% accuracy with it [3]. Ikuno et al. developed a JSL recognition system by collecting the JSL dataset using a smartphone [23]. They extracted skeleton data from the RGB image and extracted features from the skeleton information, and finally, they reported 70% accuracy with the random forest algorithm. The main drawback of the mentioned JSL recognition system is that it achieves lower performance accuracy for JSL recognition and is not satisfactory for real-life applications. Ito et al. developed a JSL recognition system by gathering all images into a single image using CNN and extracting features from the generated image based on the maximum difference from different blocks [2]. Finally, they employed an SVM for multi-class classification and achieved 84.20% accuracy with ten JSL words-based datasets. Kwolek et al. developed another RGB image-based JSL recognition system where they followed the image generation, feature extraction and classification technique [24]. They first used the Generative Adversarial Network (GAN) technique to increase the training image by generating the synthetic data with the graph technique, and then they thought about skin segmentation using ResNet34. Finally, they used an ensemble technique for the classification, which achieved 92.01% accuracy. Although this method reported higher performance accuracy than the existing method, the GAN technique for image generation leads to high computational complexity. To deploy the method, it is important to ensure the system's high-performance accuracy, efficiency, and generalizability. We proposed a hand skeleton-based feature extraction and classification method for developing a JSL recognition system to overcome the problem.

## 3 Dataset Description

Two datasets have been utilized in this research. Also, JSL is one of the most famous languages in the world.

### *3.1 Our Lab JSL Dataset*

Therefore, to utilize the hand joint estimation technique, we built a new dataset that contains the same 41 Japanese sign characters that have been utilized in the public dataset. However, in this new dataset, the size of the samples has been kept to 400 $*$ 400. In later sections, it has been shown that due to the usage of such images, the hand joint estimation was easier and therefore, higher performance was achieved. Fig. 1 illustrates the example of input images and estimated hand joints drawn on the input images for the new dataset. In this new dataset, there are 7380 images containing 180 samples per class. The images were captured from 18 individuals, with ten images per person. Table 1 showcases the distribution of each character in the JSL public dataset and the JSL new dataset. The signs の (no), も (mo), り (ri), を (wo), and ん (n) cannot be acquired with an RGB camera because they involve movement.

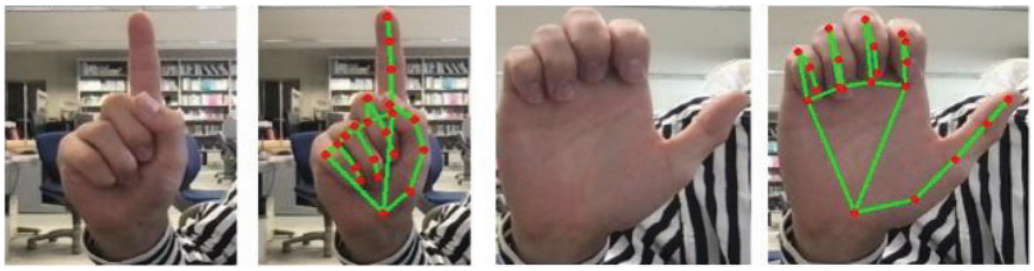
**Figure 1:** Example of input images and corresponding estimated hand joints for the new dataset

**Table 1:** The number of JSL our collected dataset

| Signs | No. of samples in our dataset | Signs | No. of samples in our datase |
|---|---|---|---|
| あ(a) | 180 | に(ni) | 180 |
| い(i) | 180 | ぬ(nu) | 180 |
| う(u) | 180 | ね(ne) | 180 |
| え(e) | 180 | は(ha) | 180 |
| お(o) | 180 | ひ(hi) | 180 |
| か(ka) | 180 | ふ(fu) | 180 |
| き(ki) | 180 | へ(he) | 180 |
| く(ku) | 180 | ほ(ho) | 180 |
| け(ke) | 180 | ま(ma) | 180 |
| こ(ko) | 180 | み(mi) | 180 |
| さ(sa) | 180 | む(mu) | 180 |
| し(shi) | 180 | め(me) | 180 |
| す(su) | 180 | や(ya) | 180 |
| せ(se) | 180 | ゆ(yu) | 180 |
| そ(so) | 180 | よ(yo) | 180 |
| た(ta) | 180 | ら(ra) | 180 |
| ち(chi) | 180 | る(ru) | 180 |
| つ(tsu) | 180 | れ(re) | 180 |
| て(te) | 180 | ろ(ro) | 180 |
| と(to) | 180 | わ(wa) | 180 |
| な(na) | 180 | | |

### *3.2 Arabic Sign Language Dataset*

The dataset is collected by 50 signers (volunteers), 27 males and 23 females. Among our signers, the oldest was 69, and the youngest was 14, where the average age was 27.98 with a standard deviation of 13.99. Fig. 2 shows a sample image for all 32 signs. We instructed the signers to capture nine images for each sign using their smartphones, with three shots with different angles for each distance, i.e., close, medium, and far. Thus, 288 images of each signer were collected; however, several unqualified images were removed during the dataset's creation. Signers were free to use any hand to perform the signs. Due to different smartphone camera configurations, some images were non-square size. Those images were padded with white pixels; thus, all the images were resized to a square with a resolution of 416 × 416 pixels. Finally, our proposed dataset, ArSL21L, containing 14202 images of 32 signs with a wide range of backgrounds, is annotated with bounding boxes in PASCAL VOC [3]; format using the LabelImg program [25]. Fig. 2 demonstrates the example of an Arabic image [12,25].

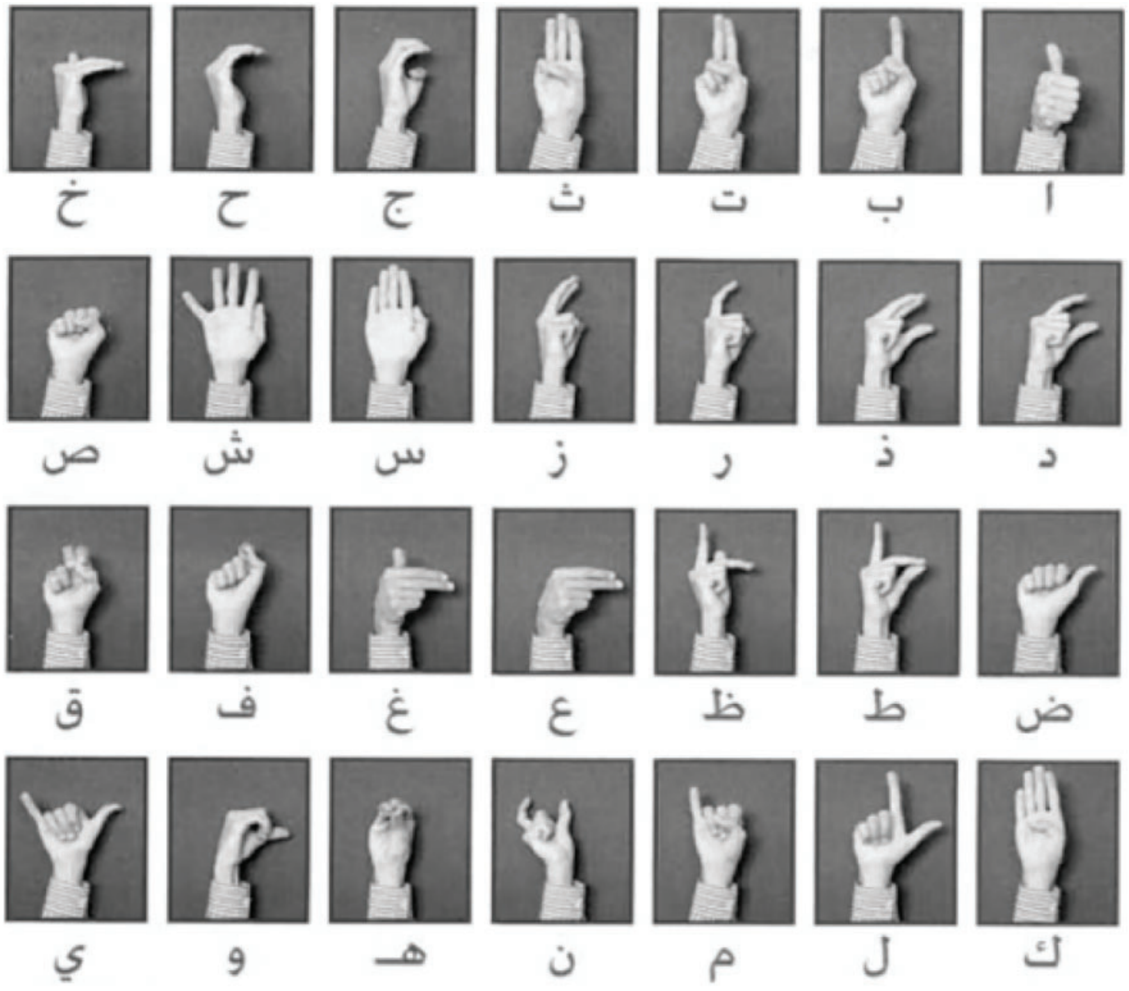

**Figure 2:** Arabic dataset examples [12]

## 4 Proposed Methodology

Fig. 3 demonstrates the proposed method architecture. In the study, we revolved around the fusion of pixel-based deep-learning features and joint skeleton-based handcrafted features for JSL recognition. To begin, we extracted joint skeleton-based features from sign language gestures, capturing intricate details of hand and body movements. These features were crucial in capturing the nuances of JSL articulation and expression.

Concurrently, we harnessed the power of CNNs to extract pixel-based features from video frames containing sign language gestures. This approach allowed our model to learn hierarchical representations of these gestures, enriching the feature set with spatial information. Integrating the hand-crafted feature with the deep learning model proved excellent in various domains and many tasks [25–27]. It can be impactful when the handcrafted features provide crucial domain-specific knowledge that might not be straightforward for the deep learning model. Hand gesture recognition using skeleton datasets can greatly benefit from the combining hand-crafted features with deep learning. The skeleton data provides joint locations, which are essentially spatial coordinates representing parts of the hand or body. This structured nature of the data offers the perfect opportunity to compute hand-crafted features. The fusion of these two distinct branch feature sets took place at multiple levels within our JSL recognition system. By combining joint skeleton-based features with pixel-based CNN features, we aimed to create a holistic and robust representation of sign language gestures. This fusion strategy capitalizes on the interpretability of handcrafted features and the discriminative capabilities of deep learning, achieving a synergistic effect. After concatenating the feature, we fed it into the feature reduction approach to select the effective features by discarding the less relevant features. Finally, we employed SVM with various kernel functions for the classification.

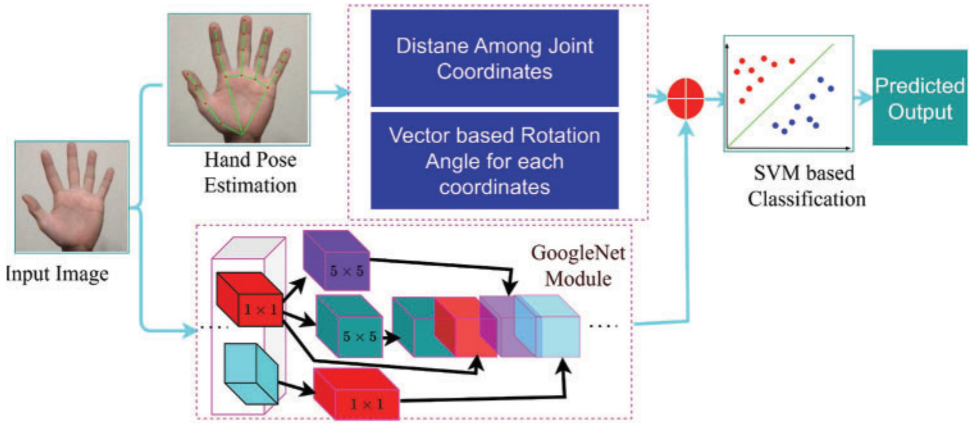

**Figure 3:** Proposed method architecture

### *4.1 Joint Estimator with Media Pipe*

Media pipe estimates hand joints using a machine-learning model that analyzes input data from a single RGB camera. The model first detects the presence of a hand in the camera image and then estimates the 3D position of the hand joints. It does this by using a complex series of mathematical calculations based on the input data, including the position and orientation of the hand, as well as other factors such as lighting and camera angle. The model then outputs a set of coordinates representing the estimated positions of each hand joint in 3D space. In the first branch, we used Mediapipe to extract the hand joint skeleton from the JSL hand gesture RGB dataset. Fig. 4 illustrates the results after applying the Media pipe.

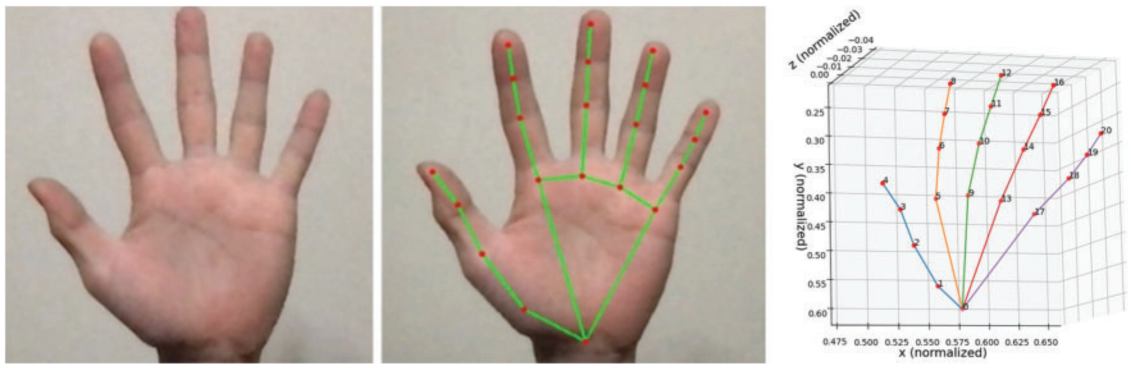

**Figure 4:** Hand pose estimation using media pipe

### *4.2 Feature Extraction*

Extracting handcrafted features from joint skeleton points is essential for the recognition of Japanese Sign Language (JSL). It ensures precise gesture capture, critical for distinguishing signs with similar handshapes. Handcrafted features offer interpretability, aiding linguistic analysis. They reduce data dimensionality, improving computational efficiency. They also enhance robustness, as they can be invariant to certain variations. Combined with other feature types, they provide comprehensive sign representations, boosting recognition accuracy. Moreover, handcrafted features can be customized to JSL's linguistic and cultural context, making recognition systems more effective and inclusive for the Deaf and hard-of-hearing communities. In the study, after extracting the 21 joint coordinates by including x, y, and z coordinates, we capture the hand-crafted features to avoid the hand position-related limitations. This feature is also important as the output value would differ even if the hand had the same signature but was placed on the left or right side of the camera. Additionally, the same hand shape can represent different characters based on their inclination in some Japanese Sign Languages. Thus, extracting features that work effectively even in such cases was necessary. To achieve this features for the distance between joint coordinates and the angle between joint coordinates were extracted.

#### *4.2.1 Distance-Based Feature Extraction*

To extract features unaffected by the hand's position in the image, distances between the 21 coordinates were calculated. The distances between neighbouring joints were excluded since they are always constant. This resulted in 190 features being obtained from each image. Using the distances between these joints, the same position can be expected even if the position of the hand changes. This is because the distances between the joints are the same no matter where they are on the screen. The formula for calculating distances is given below Eq. (1):

$$Distance = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

Here, x = Relative distance in x-coordinate between two joints

y = Relative distance in y-coordinate between two joints

z = Relative distance in z-coordinate between two joints

Fig. 5 illustrates an example of extracting distance-based features. Table 2 demonstrates the different types of distance features calculating initial and end joints. A diverse number of distances can be calculated from each of the skeleton joint points, and only 20 and 21 no skeleton joints remain empty for producing distance. In summer, we calculated different distance features from all the joints except joint numbers 20 and 21, which gave no features as distance. The main reason for the empty for generating distance is that other joints' distance calculation already covers it.

#### *4.2.2 Angle-Based Feature Extraction*

The feature value of how much the hand is tilted was calculated by calculating the direction vectors between the coordinates of each joint and then determining how much each vector was tilted from the XYZ direction. A total of 210 vectors can be created since the number of joints to be estimated is 21, and three types of XYZ data can be calculated for each vector. This resulted in 630 features related to angles. These features can increase the recognition rate of signs with the same shape but with different meanings depending on the inclination of the hand, such as the Japanese Sign Language characters for な(na), に(ni), ま(ma), and み(mi). The angle between the vectors was calculated by taking Cos values from the two space vectors and using the direction vector and the vectors in the x, y, and z-axis directions.

$$cos(\theta) = \frac{xy + yz + zx}{\sqrt{x^2 + y^2 + z^2} \cdot \sqrt{y^2 + z^2 + x^2}} \tag{2}$$

Table 3 demonstrates the all angle calculation joint vector scenarios, which generated 630 diverse angle-based features. This style also has similarities, like the distance calculation where 21 no joint set is described as empty. Similarly to distance-based features, it can be noticed that for the joint point number 21, the set is empty. This is because the earlier joint points have already covered the expected pairs. In Table 3, angle calculation of initial point and set of vectors when each joint point can be considered the endpoint.

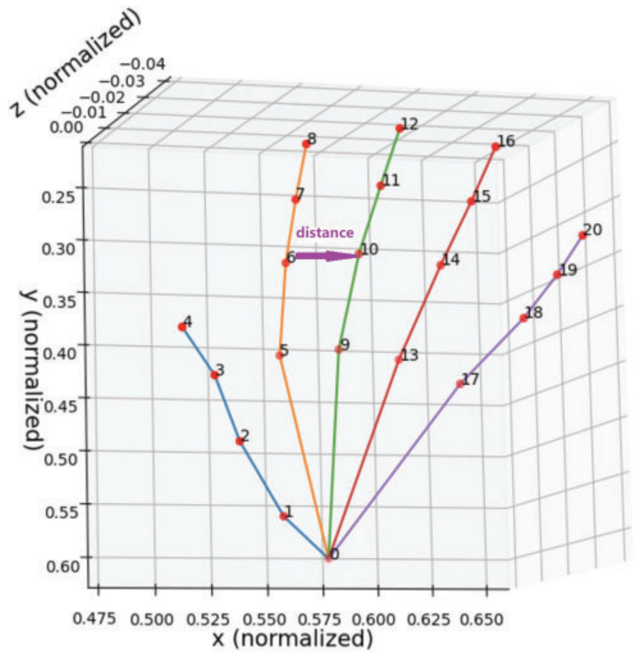

**Figure 5:** Example of extracting distance-based feature

**Table 2:** Distance calculation initial point and end points

| Starting joint number | Distance to the joint numbers | Number of distance based features |
|---|---|---|
| 1 | {3,4,6,7,8,10,11,12,14,15,16,18,19,20,21} | 15 |
| 2 | {4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21} | 18 |
| 3 | {5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20} | 17 |
| 4 | {6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21} | 16 |
| 5 | {6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21} | 16 |
| 6 | {8,9,10,11,12,13,14,15,16,17,18,19,20,21} | 14 |
| 7 | {9,10,11,12,13,14,15,16,17,18,19,20,21} | 13 |
| 8 | {10,11,12,13,14,15,16,17,18,19,20,21} | 12 |
| 9 | {10,11,12,13,14,15,16,17,18,19,20,21} | 12 |
| 10 | {12,13,14,15,16,17,18,19,20,21} | 10 |
| 11 | {13,14,15,16,17,18,19,20,21} | 9 |

(Continued)

**Table 2 (continued)**

| Starting joint number | Distance to the joint numbers | Number of distance based features |
|---|---|---|
| 12 | {14,15,16,17,18,19,20,21} | 8 |
| 13 | {14,15,16,17,18,19,20,21} | 8 |
| 14 | {16,17,18,19,20,21} | 6 |
| 15 | {16,17,18,19,20,21} | 5 |
| 16 | {17,18,19,20,21} | 4 |
| 17 | {18,19,20,21} | 4 |
| 18 | {20,21} | 2 |
| 19 | {21} | 1 |
| 20 | {} | 0 |
| 21 | {} | 0 |

**Table 3:** Angle calculation initial point and set of vectors when each joint point can be considered the endpoint

| Joint index | Starting joint (P) Variable joint (V) | Other joints ( V) | Features |
|---|---|---|---|
| 0 | $\overrightarrow{P_0V}$ | $P_1, P_2, P_3, P_4, P_5, P_6, P_7,$ $P_8, P_9, P_{10}, P_{11}P_{12}, P_{13}, P_{14},$ $P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$ | $20 \times 3$ |
| 1 | $\overrightarrow{P_1V}$ | $P_2, P_3, P_4, P_5, P_6, P_7$ $P_8, P_9, P_{10}, P_{11}P_{12}, P_{13}, P_{14}$ $P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$ | $19 \times 3$ |
| 2 | $\overrightarrow{P_2V}$ | $P_3, P_4, P_5, P_6, P_7$ $P_8, P_9, P_{10}, P_{11}P_{12}, P_{13}, P_{14}$ $P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$ | $18 \times 3$ |
| 3 | $\overrightarrow{P_3V}$ | $P_4, P_5, P_6, P_7$ $P_8, P_9, P_{10}, P_{11}P_{12}, P_{13}, P_{14}$ $P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$ | $17 \times 3$ |
| 4 | $\overrightarrow{P_4V}$ | $P_5, P_6, P_7$ $P_8, P_9, P_{10}, P_{11}P_{12}, P_{13}, P_{14}$ $P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$ | $16 \times 3$ |
| 5 | $\overrightarrow{P_5V}$ | $P_5, P_6, P_7$ $P_8, P_9, P_{10}, P_{11}P_{12}, P_{13}, P_{14}$ $P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$ | $15 \times 3$ |

(Continued)

**Table 3 (continued)**

| Joint index | Starting joint (P) Variable joint (V) | Other joints ( V) | Features |
|---|---|---|---|
| 6 | $\overrightarrow{P_6 V}$ | $P_7, P_8, P_9, P_{10}, P_{11} P_{12}, P_{13}$ $P_{14}$ $P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$ .. | $14 \times 3$ .. |
| 19 | $\overrightarrow{P_{19} V}$ | $P_{20}$ | $1 \times 3$ |
| 20 | $\overrightarrow{P_{20} V}$ | | 0 |

### *4.3 GoogleNet Features*

In the second stream, we extracted deep learning features using the pre-trained GoogleNet model from the RGB images in the JSL dataset. Researchers have developed many deep learning-based transfer learning models to extract the effective features to overcome the inefficient dataset [28–31]. In addition, BenSignNet [32], CNN [33], DenseNetwork [34], Deep Residual Model [35], VeryDeepCNN [36], ImageNet [37] and Invert residual network [38] also used by many researchers to extract effective feature and classification to implement the vision based hand gesture recognition system. We did this by removing the final classification layers and keeping the feature extraction layers intact. Pass each image through the modified model, and the output from one of the intermediate layers can be considered as the extracted features. Machine learning has offered a wide range of techniques and Application-based models and huge data sizes for processing and problem-solving. Deep learning, a subset of Machine Learning, is usually more complex. So, thousands of images are needed to get accurate results. The GoogleNet architecture was proposed by researchers from Google in 2014. The research paper was titled "Going Deeper with Convolutions". This architecture was a winner at the ILSVRC 2014 classification of image challenge [39]. It has shown a significant decrease in error rate if we compare it with the previous winner, AlexNet. GoogleNet architecture uses techniques such as $1 \times 1$ convolutions in the middle of the architecture and global average pooling that is shown in the second stream of Fig. 3 that is different from the AlexNet architecture. Using $1 \times 1$ convolution as intermediate and global average pooling creates a deeper architecture. This architecture is 22 layers deep. Google Net uses $1 \times 1$ convolutions in architecture. The convolutions are used to decrease the number of parameters (weights and biases) of the architecture. For a convolution, when the filter number is 48 but the filter size is $5 \times 5$, the number of computations can be written as $(14 \times 14 \times 48) \times (5 \times 5 \times 480) = 112.9$ M. On the other side, for the $1 \times 1$ convolution with 16 filters, the number of computations can be written as. $(14 \times 14 \times 16) \times (1 \times 1 \times 480) + (14 \times 14 \times 48) \times (5 \times 5 \times 16) = 5.3$ M. The inception architecture in GooLeNet has some intermediate classifier branches in the middle of the architecture. These branches are used only while training. These branches consist of a $5 \times 5$ average pooling layer with a stride of 3, one $1 \times 1$ convolution with 128 filters, and two fully connected layers of 1024 outputs. The architectural details can be described in the following steps: (a) A $1 \times 1$ convolution with 128 filters for dimension reduction and rectified linear unit (ReLU) activation. (b) An average polling layer of filter size 55 and stride is three values. (c) Dropout regularization with dropout ratio $= 0.7$. (d) A fully connected layer with 1024 outputs and ReLU activation. We did not use any classification module, but we used the output of the GoogleNet architecture as a feature of the second stream, and the feature dimension is 1024, which is concatenated with the handcrafted feature to produce the final feature.

## 4.4 Feature Reduction

We got 819 features from the hand-crafted first stream and 1024 from the deep learning-based second stream. After concatenating the two streams, we got 1843 features in total. High-dimensional features can carry the effective information of the JSL gesture, but they may carry some irrelevant features, which will lead to a reduction in computational complexity. Selection of the potential feature and discarding the irrelevant feature may improve the performance accuracy and efficiency of the system. Feature selection is known as a critical step in the process of building machine learning models, as it helps improve model performance, reduces overfitting, and speeds up training. Here, we used a popular method for feature selection, namely the Boruta algorithm, which leverages the power of Random Forest to identify the most relevant features for a given problem.

### 4.4.1 The Boruta Algorithm

Boruta, inspired by the spirit of Darwinian evolution, mimics the competitive world of features to select those that truly matter. The procedure begins with duplicating the original dataset to create a shadow dataset, where the features' order is randomized. Next, a Random Forest classifier is trained on the combined dataset, encompassing the original and shadow features. During the algorithm's iterations, Boruta assesses the importance of each feature by comparing it with the performance of its corresponding shadow feature. Features that consistently outperform their shadow counterparts are deemed significant and retained. On the other hand, those features that fail to exhibit consistent superiority are rejected. The algorithm concludes when all features prove their significance or are eliminated from contention [19].

### 4.4.2 Random Forest's Role

The strength of Boruta lies in its partnership with Random Forest, a versatile ensemble learning technique. Random Forest constructs a multitude of decision trees to make predictions, each tree emphasizing different subsets of features. By comparing a feature's performance against its shadow counterpart across multiple trees, Boruta mitigates the potential of overfitting, enhancing its reliability. There are various advantages of the feature reduction method, such as reduced overfitting, comprehensive feature assessment, efficient computation, and robust performance [19].

## 4.5 Classification with SVM

After selecting the potential features using the Boruta algorithm approach, we employed the conventional machine learning model SVM with various kernel approaches for recognition and prediction. SVM is a powerful machine learning algorithm mainly used for classification and regression analysis. The main objective of SVM is to find a hyperplane that separates data points of different classes in a high-dimensional space. In classification, SVM determines the best boundary between two classes by maximizing the margin between the two closest points of the different classes. SVM is especially effective in handling complex, non-linear datasets, as it can project data to a higher-dimensional space where a linear hyperplane can separate them. In the study, we used several kernel functions, such as polynomial, radial basis function, and sigmoid functions, aiming to transform the data into a higher-dimensional space to make it more separable. Also, the SVM was originally designed for binary classification tasks, but it can be extended to handle multi-class classification problems like classifying the 41 labels of Japanese Sign Language (JSL) through various techniques. In the study, we did this using the One-*vs*-Rest (OvR) Approach, where in the OvR approach, we train 41 separate binary classifiers, each distinguishing one JSL sign from the rest. This means you have a binary classifier for each sign (41 classifiers). During prediction, we run all 41 classifiers on an input, and the sign corresponding to the classifier with the highest confidence score is chosen as the predicted sign [13].

## 5 Experimental Analysis and Results

We described here the accuracy of the proposed model after evaluating both the existing dataset and our JSL dataset. In this study, we employed two branches of features: the first branch consisted of handcrafted features, including distance and angle, while the second branch included deep learning-based features. Additionally, we utilized augmentation techniques, such as Random Affine transformations ranging from 0.1 to 0.1 and 0.95 to 1.05, rotations between −20 degrees and 20 degrees, a 10 per cent shift in both vertical and horizontal directions, and scale within the range of 0.95 to 1.05 per cent.

We conducted experiments with three different configurations: a stream based solely on handcrafted features, a stream based solely on deep learning features, and a combined stream that included augmentation and one without augmentation.

### *5.1 Experimental Setting*

We implemented the proposed model in Python with TensorFlow, pandas, NumPy, and sci-kit-learn Python packages to test the proposed model. To experiment with the system, we used a CPU machine with 16 GB RAM. In this research, both user-dependent and user-independent SVM have been applied. In user-dependent SVM, a random 70% of the datasets were utilized for training, and the rest of the 30% data were kept for testing. However, in user-independent SVM, data from 17 people was used for training and data from 1 person was used for testing. In both cases, the parameters $C$ and $\gamma$ are adjusted during training, and the best one is applied to the test set.

### *5.2 Experimental Result with JSL Dataset*

Table 4 shows the accuracy score of the proposed model, including the number of features, selected features, and the kernel of the SVM, along with their corresponding performance accuracies. The table displays the performance accuracy for three distinct configurations. The first configuration consists of a stream that relies exclusively on handcrafted features, achieving an accuracy of 93.70%. The second configuration is based solely on deep learning features and achieves an accuracy of 90.08%. In contrast, the third configuration is a combined stream that incorporates augmentation techniques and attains an accuracy of 96.30%. Additionally, there is a fourth configuration, which is identical to the combined stream but without augmentation, achieving a remarkable accuracy of 98.53%.

**Table 4:** Performance results proposed with the lab JSL dataset

| Model branch | Augmentation | Hand crafted | Deep learning | Total | Selected | SVM | Performance |
|---|---|---|---|---|---|---|---|
| | | Feature | Feature | Feature | Feature | Parameters | Accuracy [%] |
| Hand crafted features branch | N/A | 819 | N/A | N/A | 819 | $C = 1500$, $\gamma = 0.3$ | 93.70 |
| CNN branch | N/A | N/A | 1024 | N/A | 1024 | $C = 1500$, $\gamma = 0.3$ | 90.08 |
| Two branches | Random affine rotation shifting scaling | 819 | 1024 | 1843 | 777 | $C = 1500$, $\gamma = 0.3$ | 96.30 |
| Two branches | No augmentation | 819 | 1024 | 1843 | 777 | $C = 1500$, $\gamma = 0.3$ | **98.53** |

As mentioned earlier, for the JSL dataset, we extracted 819 handcrafted features based on the distance and angle. Subsequently, we obtained 1024 features using the deep learning-based method, and an additional 1843 features were generated by combining all feature extraction methods. To identify the most relevant features, we employed the Boruta and random forest methods, which resulted in the selection of 777 potential features from the pool of 1843 features. Ultimately, the selected features were fed into the machine learning-based SVM classifier. The study yielded a remarkable accuracy of 98.53% through the use of SVM with hyperparameter tuning.

Table 5 shows the label-wise precision-recall and F1-score where. It meticulously outlines precision, recall, and F1-score for each label, offering insight into the model's effectiveness in correctly identifying positive instances and minimizing false negatives. Examining the results reveals remarkable performance across multiple fronts. Several labels, such as 0, 3, 4, 5, 11, 17, 19, and 40, exhibit flawless precision, recall, and F1-scores at 100.00%. This suggests an exceptional ability to predict instances belonging to these categories accurately. Furthermore, various labels manifest high precision rates coupled with near-perfect recall, indicating a propensity not only to predict positives correctly but also to effectively capture nearly all actual positive instances effectively. Notably, label 24 achieves a perfect 100.00% across all metrics, highlighting the model's ability to harmoniously attain precision, recall, and F1-scores harmoniously. Throughout the table, the F1 scores reflect the balance between precision and recall, illuminating the model's overall effectiveness by considering both false positives and false negatives. The weighted nature of the F1-score ensures a comprehensive evaluation, especially for imbalanced datasets where certain labels might be underrepresented. In summary, the table encapsulates the classification model's prowess in a granular manner, depicting its capacity to discern intricate patterns across diverse labels. The stellar performance in precision, recall, and F1-scores for numerous labels underscores the model's proficiency in accurate classification and thorough identification of relevant instances. While some labels exhibit slightly lower metrics, the overall results testify to the model's robustness and efficacy in handling a multifaceted classification task. Fig. 6 shows the confusion matrix for the proposed JSL dataset.

**Table 5:** Precision recall and F1-score of the proposed JSL dataset

| Label | Precision (%) | Recall (%) | F1-score (%) | Label | Precision (%) | Recall (%) | F1-score (%) |
|---|---|---|---|---|---|---|---|
| 0 | 100.00 | 100.00 | 100.00 | 21 | 98.88 | 98.32 | 98.60 |
| 1 | 100.00 | 99.44 | 99.72 | 22 | 98.33 | 98.88 | 98.61 |
| 2 | 98.88 | 98.88 | 98.88 | 23 | 100.00 | 99.44 | 99.72 |
| 3 | 100.00 | 100.00 | 100.00 | 24 | 100.00 | 100.00 | 100.00 |
| 4 | 100.00 | 100.00 | 100.00 | 25 | 99.44 | 100.00 | 99.72 |
| 5 | 100.00 | 100.00 | 100.00 | 26 | 99.43 | 98.31 | 98.87 |
| 6 | 99.44 | 99.44 | 99.44 | 27 | 100.00 | 99.43 | 99.71 |
| 7 | 99.44 | 100.00 | 99.72 | 28 | 100.00 | 99.44 | 99.72 |
| 8 | 99.44 | 98.32 | 98.88 | 29 | 99.44 | 98.88 | 99.16 |
| 9 | 100.00 | 99.41 | 99.71 | 30 | 99.43 | 97.74 | 98.58 |
| 10 | 98.90 | 100.00 | 99.44 | 31 | 100.00 | 100.00 | 100.00 |
| 11 | 100.00 | 100.00 | 100.00 | 32 | 99.44 | 99.44 | 99.44 |
| 12 | 97.79 | 100.00 | 98.88 | 33 | 100.00 | 100.00 | 100.00 |
| 13 | 100.00 | 98.88 | 99.44 | 34 | 98.89 | 99.44 | 99.16 |
| 14 | 97.22 | 98.87 | 98.04 | 35 | 98.35 | 100.00 | 99.17 |
| 15 | 99.44 | 100.00 | 99.72 | 36 | 98.33 | 98.88 | 98.61 |

(Continued)

**Table 5 (continued)**

| Label | Precision (%) | Recall (%) | F1-score (%) | Label | Precision (%) | Recall (%) | F1-score (%) |
|---|---|---|---|---|---|---|---|
| 16 | 98.88 | 98.32 | 98.60 | 37 | 99.44 | 99.44 | 99.44 |
| 17 | 99.44 | 99.44 | 99.44 | 38 | 100.00 | 100.00 | 100.00 |
| 18 | 98.35 | 100.00 | 99.17 | 39 | 98.82 | 96.53 | 97.66 |
| 19 | 99.44 | 99.44 | 99.44 | 40 | 100.00 | 100.00 | 100.00 |
| 20 | 98.30 | 98.30 | 98.30 | | | | |

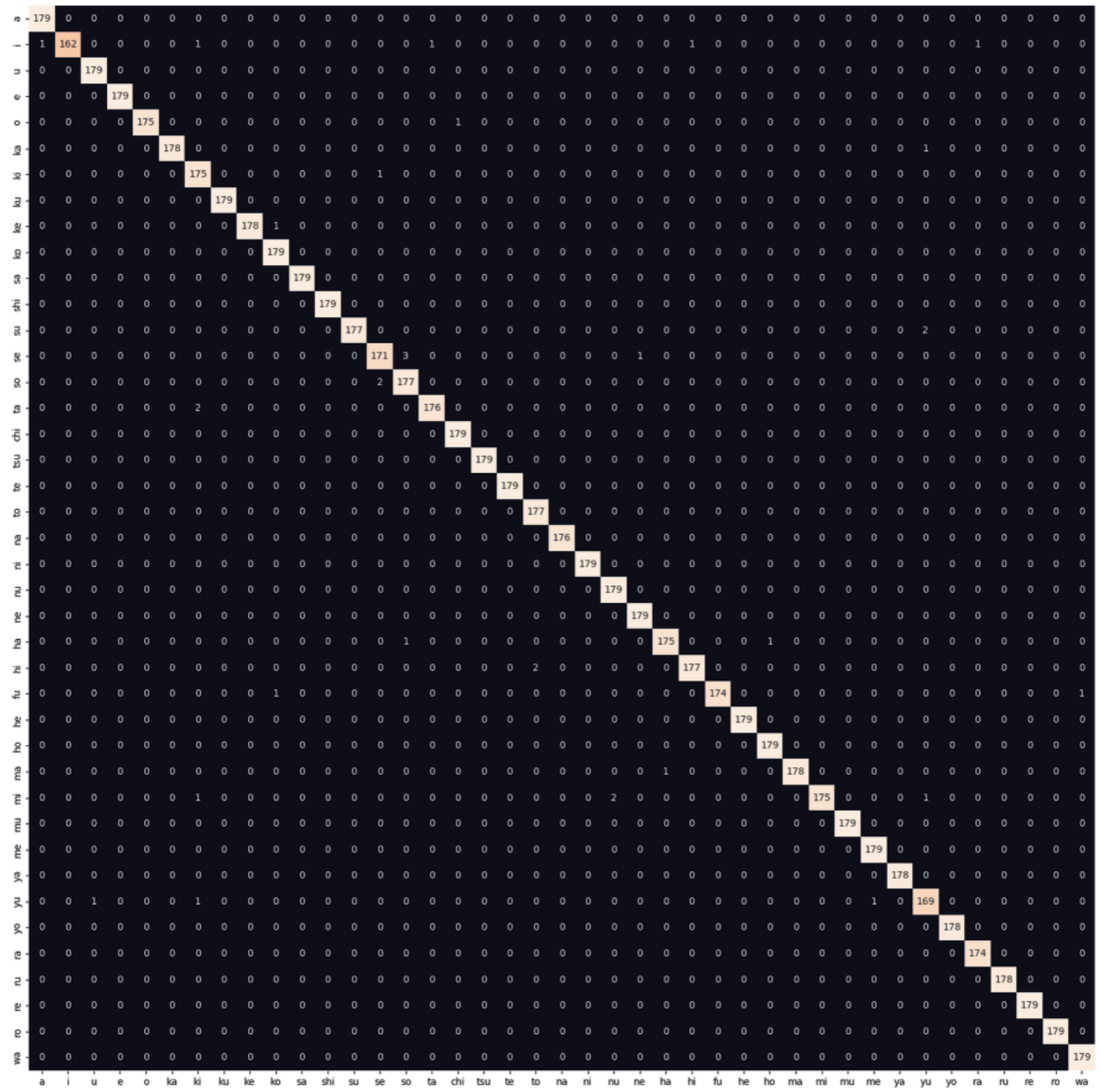

**Figure 6:** Confusion matrix for our lab JSL dataset

### *5.3 Experimental Result with a Benchmark Public Dataset*

Table 6 presents a detailed breakdown of precision, recall, and F1-score metrics for the Arabic datasets. Noteworthy observations include exceptional scores for some classes: Class 1 achieves a perfect 100.00% in all metrics, and Class 15 attains 100.00% precision and recall. While some classes exhibit slightly lower F1-scores, the model demonstrates strong performance overall. A few classes, like class 18, show a precision-recall imbalance, while others, such as class 20, showcase well-balanced performance. The results underscore the model's effectiveness in classification, with varying degrees of accuracy and completeness across different classes. Fig. 7 shows the confusion matrix of the proposed model.

**Table 6:** Precision Recall and F1-score of the Arabic dataset

| Label | Precision | Recall | F1-score | Label | Precision | Recall | F1-score |
|---|---|---|---|---|---|---|---|
| 0 | 96.88 | 99.20 | 98.02 | 15 | 100.00 | 100.00 | 100.00 |
| 1 | 100.00 | 100.00 | 100.00 | 16 | 91.04 | 93.13 | 92.08 |
| 2 | 99.24 | 97.76 | 98.50 | 17 | 99.25 | 98.51 | 98.88 |
| 3 | 93.98 | 95.42 | 94.70 | 18 | 86.72 | 92.50 | 89.52 |
| 4 | 90.91 | 90.09 | 90.50 | 19 | 100.00 | 90.98 | 95.28 |
| 5 | 96.30 | 97.01 | 96.65 | 20 | 97.73 | 96.27 | 96.99 |
| 6 | 96.03 | 94.53 | 95.28 | 21 | 94.93 | 97.04 | 95.97 |
| 7 | 89.21 | 93.23 | 91.18 | 22 | 98.54 | 100.00 | 99.26 |
| 8 | 92.91 | 92.91 | 92.91 | 23 | 94.81 | 96.24 | 95.52 |
| 9 | 100.00 | 100.00 | 100.00 | 24 | 98.44 | 93.33 | 95.82 |
| 10 | 96.12 | 97.64 | 96.88 | 25 | 100.00 | 94.89 | 97.38 |
| 11 | 95.20 | 93.70 | 94.44 | 26 | 93.43 | 94.81 | 94.12 |
| 12 | 98.48 | 96.30 | 97.38 | 27 | 99.21 | 95.45 | 97.30 |
| 13 | 97.01 | 96.30 | 96.65 | 28 | 94.31 | 95.87 | 95.08 |
| 14 | 96.38 | 98.52 | 97.44 | 29 | 94.85 | 97.73 | 96.27 |

Table 7 gives the accuracy score of the proposed model and the state-of-the-art comparison where the proposed model achieved 95.84% accuracy. Compared to the other state-of-the-art models, ResNet1 transfer learning and ResNet2 transferring gave 35.00% and 91.04% accuracy, respectively. In addition, another researcher employed a based approach and gave 92.00% accuracy. In summary, we see that our proposed model achieved higher than all the previous methods. The table provides a concise overview of different models' performance on the "ArSL21L Arabic" dataset, highlighting their respective features, feature selection, SVM parameters, and resulting performance accuracy. The first three models, labelled ResNet1, ResNet2, and CNN, have unspecified handcrafted and deep learning features, total features, selected features, and SVM parameters. ResNet1 achieves a performance accuracy of 35.00%, while ResNet2 and CNN achieve significantly higher accuracies of 91.04% and 92.00%, respectively. The proposed model, tailored for the same "ArSL21L Arabic" dataset, employs both handcrafted features (819) and deep learning features (1024), combining to form a total of 1843 features. After feature selection, 820 features are chosen. The SVM parameters $C = 100$ and $\gamma = 0.0005$ are utilized. This model demonstrates an impressive performance accuracy of 95.84%, surpassing the other mentioned models. Overall, the table underlines the efficacy of the proposed model by showcasing its ability to leverage a combination of handcrafted and deep learning features, resulting in a considerably high accuracy on the specified dataset when compared to alternative approaches.

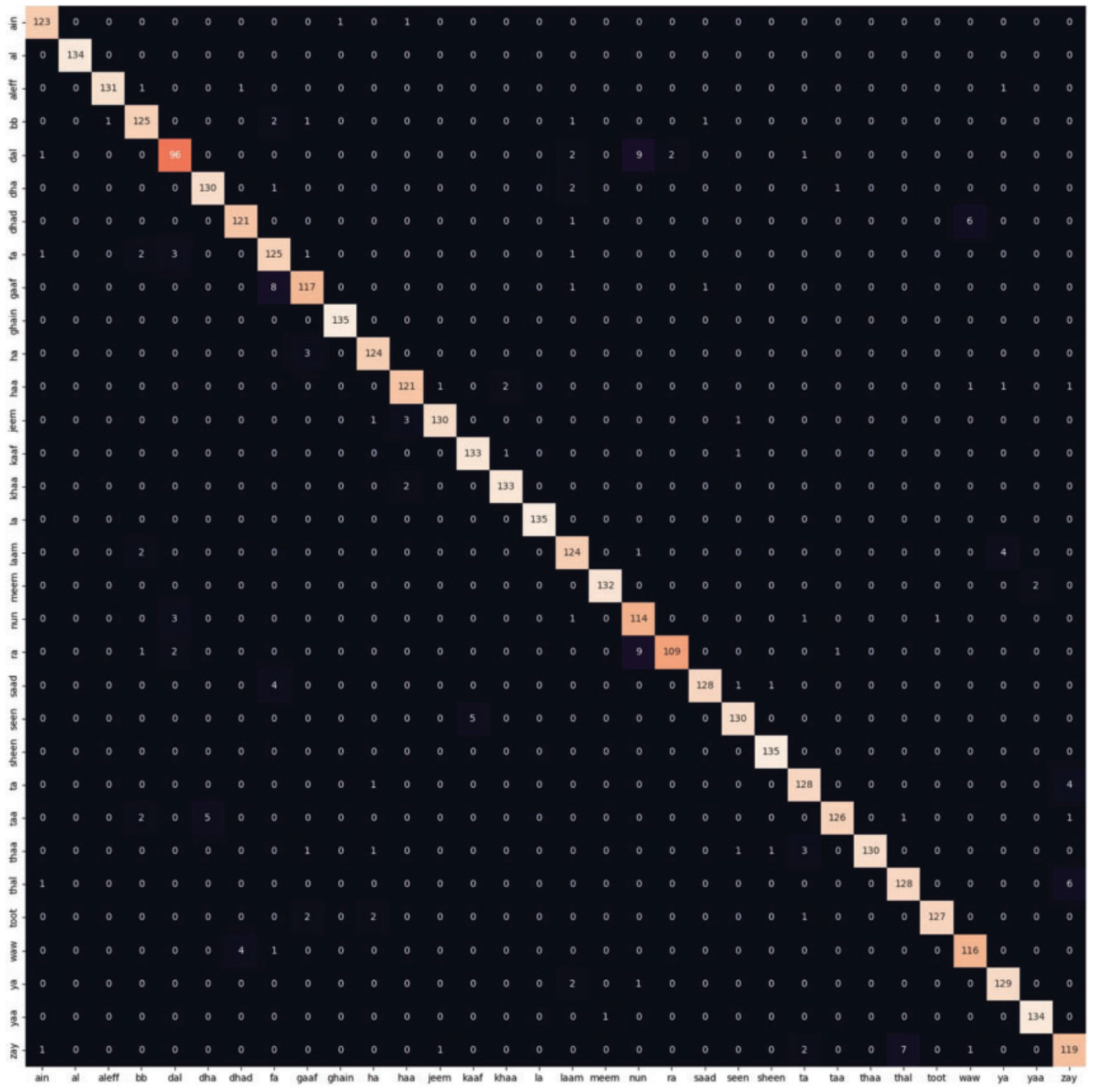

**Figure 7:** Confusion matrix for Arabic dataset

**Table 7:** Performance results proposed with the public Arabic dataset and state-of-the-art comparison

| Model name | Dataset name | Hand crafted Feature | Deep learning Feature | Total Feature | Selected Feature | SVM Parameters | Performance Accuracy (%) |
|---|---|---|---|---|---|---|---|
| ResNet1 [25] | ArSL21L Arabic | n/a | n/a | n/a | n/a | n/a | 35.00 |
| ResNet2 [25] | ArSL21L Arabic | n/a | n/a | n/a | n/a | n/a | 91.04 |
| CNN [25] | ArSL21L Arabic | n/a | n/a | n/a | n/a | n/a | 92.00 |
| Proposed Model | ArSL21L Arabic | 819 | 1024 | 1843 | 820 | $C = 100$, $\gamma = 0.0005$ | 95.84 |

These results demonstrate the significant impact of augmentation techniques on the model's performance. The configuration without augmentation achieves the highest accuracy, indicating that the model can make accurate predictions solely based on the inherent features of the data. On the other hand, the combined stream, which includes augmentation, still performs exceptionally well and suggests that the augmentation techniques help improve the model's ability to generalize and handle variations in the data. These findings highlight the versatility of the model and its adaptability to different feature sets and preprocessing strategies. Further analysis can explore the specific ways in which augmentation techniques contribute to performance improvements and whether they have varying effects on handcrafted features *vs*. deep learning features.

## 6 Conclusions

Our study introduces an innovative approach to Japanese Sign Language (JSL) recognition, leveraging the strengths of joint skeleton-based handcrafted features and pixel-based Convolutional Neural Network (CNN) features. This fusion not only significantly enhances recognition accuracy but also contributes to improving the accessibility and inclusivity of sign language communication for the Deaf and hard-of-hearing communities. The integration of these two feature streams results in a large feature dimension, leading to increased computational complexity. To manage this complexity, we employed a feature selection algorithm to identify and retain the most relevant features while discarding irrelevant ones. This dimensionality reduction approach optimized the efficiency and effectiveness of our system. Our choice of a machine learning-based classification algorithm, Support Vector Machine (SVM), as the classifier yielded high-performance accuracy. This achievement underscores the efficiency and effectiveness of the proposed system, particularly in its ability to outperform previous studies that relied on expensive devices. We successfully employed a relatively inexpensive webcam, making our system more accessible and cost-effective. Looking ahead, our future work will focus on extending our approach to word-level JSL recognition systems, further advancing the field of sign language recognition and enhancing communication accessibility for the deaf and hard-of-hearing communities.

**Acknowledgement:** The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

**Funding Statement:** This work was supported by the Competitive Research Fund of the University of Aizu, Japan.

**Author Contributions:** The authors confirm their contribution to the paper as follows: study conception and design: Jungpil Shin, Md. Al Mehedi Hasan; data collection: Abu Saleh Musa Miah, Kota Suzuki; analysis and interpretation of results: Jungpil Shin, Koki Hirooka, Abu Saleh Musa Miah; draft manuscript preparation: Md. Al Mehedi Hasan, Kota Suzuki, Koki Hirooka. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** Arabic Dataset: https://www.kaggle.com/datasets/ammarsayedtaha/arabic-sign-language-dataset-2022.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

1. Cabinet Office Japan (2023). White paper on persons with disabilities 2023. https://www8.cao.go.jp/shougai/english/annualreport/2023/pdf/index.pdf (accessed on 08/06/2023).
2. Ito, S. I, Ito, M., Fukumi, M. (2019). A method of classifying japanese sign language using gathered image generation and convolutional neural networks. *2019 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, Fukuoka, Japan.
3. Funasaka, M., Ishikawa, Y., Takata, M., Joe, K. (2015). Sign language recognition using leap motion controller. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, pp. 263–269. The Steering Committee of the World Congress in Computer Science, Computer, Vegas, Nevada, USA.
4. Ministry of Health, Labour and Welfare, Japan (2018). Survey on difficulties in living. https://www.mhlw.go.jp/toukei/list/dl/ (accessed on 08/06/2023).
5. Ohki, J. (2014). Sign language service with it for the deaf people in japan "tech for the deaf". *Journal of Information Processing and Management, 57(4),* 234–242.
6. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G. et al. (2020). Mediapipe hands: On-device real-time hand tracking. arXiv preprint arXiv:2006.10214.
7. Kobayashi, H., Ishikawa, T., Watanabe, H. (2019). Classification of japanese signed character with pose estimation and machine learning. *IEICE General ConferenceProceedings of the Conference on Information and Systems*, Hiroshima, Japan.
8. Weichert, F., Bachmann, D., Rudak, B., Fisseler, D. (2013). Analysis of the accuracy and robustness of the leap motion controller. *Sensors, 13(5),* 6380–6393.
9. Supancic III, J. S., Rogez, G., Yang, Y., Shotton, J., Ramanan, D. (2015). Depth-based hand pose estimation: Data, methods, and challenges. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Montreal, QC, Canada.
10. Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., Twombly, X. (2007). Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding, 108(1),* 52–73.
11. Tompson, J., Stein, M., Lecun, Y., Perlin, K. (2014). Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics, 33(5),* 1–10.
12. Aly, S., Osman, B., Aly, W., Saber, M. (2016). Arabic sign language fingerspelling recognition from depth and intensity images. *2016 12th International Computer Engineering Conference (ICENCO)*, Giza, Egypt, IEEE.
13. Kabir, M. H., Mahmood, S., Al Shiam, A., Musa Miah, A. S., Shin, J. et al. (2023). Investigating feature selection techniques to enhance the performance of EEG-based motor imagery tasks classification. *Mathematics, 11(8)*, 1921.
14. Miah, A. S. M., Hasan, M. A. M., Shin, J. (2023). Dynamic hand gesture recognition using multi-branch attention based graph and general deep learning model. *IEEE Access, 11,* 4703–4716.
15. Miah, A. S. M., Shin, J., Hasan, M. A. M., Rahim, M. A., Okuyama, Y. (2023). Rotation, translation and scale invariant sign word recognition using deep learning. *Computer Systems Science and Engineering, 44(3),* 2521–2536. https://doi.org/10.32604/csse.2023.029336
16. Miah, A. S. M., Hasan, M. A., Leo, B. (2001). Random forests. *Machine Learning, 45,* 5–32.
17. Shin, J., Musa Miah, A. S., Hasan, M. A. M., Hirooka, K., Suzuki, K. et al. (2023). Korean sign language recognition using transformer-based deep neural network. *Applied Sciences, 13(5),* 2049–3029.
18. Miah, A. S. M., Hasan, M. A. M., Shin, J., Okuyama, Y., Tomioka, Y. (2023). Multistage spatial attention-based neural network for hand gesture recognition. *Computers, 12(1),* 1–13.

19. Miron, B. Kursa, Witold, R. Rudnicki (2010). Feature Selection with the Boruta Package. *Journal of Statistical Software, 36(11),* 1–13.
20. Shiraishi, Y., Tsuchiya, T., Kato, N., Yoneyama, F., Shitara, M. (2020). Fingerprint recognition using multidimensional time series data using deep learning. *Tsukuba University of Technology Techno Report, 28(1),* 122–123.
21. Shah, S. A. A., Bukhari, S. S. A., Humayun, M., Jhanjhi, N., Abbas, S. F. (2019). Test case generation using unified modeling language. *2019 International Conference on Computer and Information Sciences (ICCIS)*, Sakaka, Saudi Arabia, IEEE.
22. Hosoe, H., Sako, S., Kwolek, B. (2017). Recognition of jsl finger spelling using convolutional neural networks. *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, Nagoya University, Nagoya, Japan.
23. Wang, X., Tarrío, P., Metola, E., Bernardos, A. M., Casar, J. R. (2012). Gesture recognition using mobile phone's inertial sensors. *Distributed Computing and Artificial Intelligence: 9th International Conference*, Alamanca, Spain, Springer.
24. Kwolek, B., Baczynski, W., Sako, S. (2021). Recognition of JSL fingerspelling using deep convolutional neural networks. *Neurocomputing, 456,* 586–598.
25. Batnasan, G., Gochoo, M., Otgonbold, M. E., Alnajjar, F., Shih, T. K. (2022). Arsl21l: Arabic sign language letter dataset benchmarking and an educational avatar for metaverse applications. *2022 IEEE Global Engineering Education Conference (EDUCON)*, University of Geneva, Switzerland.
26. Huang, X., Li, Z., Zhang, M., Gao, S. (2022). Fusing hand-crafted and deep-learning features in a convolutional neural network model to identify prostate cancer in pathology images. *Frontiers in Oncology, 12,* 994950.
27. Ewe, E. L. R., Lee, C. P., Kwek, L. C., Lim, K. M. (2022). Hand gesture recognition via lightweight vgg16 and ensemble classifier. *Applied Sciences, 12(15),* 7643.
28. Chollet, F. (2016). Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv: 1610.02357.
29. Szegedy, C., Ioffe, S., Vanhoucke, V. (2016). Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261.
30. He, K., Zhang, X., Ren, S., Sun, J. (2016). Identity mappings in deep residual networks. arXiv preprint arXiv:1603.05027.
31. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2015). Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567.
32. Miah, A. S. M., Shin, J., Hasan, M. A. M., Rahim, M. A. (2022). BenSignNet: Bengali sign language alphabet recognition using concatenated segmentation and convolutional neural network. *Applied Sciences, 12(8),* 3933.
33. Akash, H. S., Rahim, M. A., Miah, A. S. M., Okuyama, Y., Tomioka, Y. et al. (2023). Generalized technique for potato leaves disease classification using convolutional neural network. In: Tuba, M., Akashe, S., Joshi, A. (Eds.), *ICT systems and sustainability*. Singapore: Springer Nature Singapore.
34. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K. Q. (2016). Densely connected convolutional networks. arXiv preprint arXiv:1608.06993.
35. He, K., Zhang, X., Ren, S., Sun, J. (2015). Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385.
36. Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
37. Krizhevsky, A., Sutskever, I., Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60(6),* 84–90.

38. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L. C. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. arXiv preprint arXiv:1801.04381.
39. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. et al. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA.