Article

Developing an Intelligent Recognition System for British Sign Language: A Step Towards Inclusive Communication

Sakirulai Olufemi Isiaq [1,*] and **Shany Stephen** [2]

[1] Creative Computing Institute, University of Arts London, London WC1V 7EY, UK

[2] Department of Computing, Solent University, Southampton SO14 0YN, UK

* Correspondence: f.isiaq@arts.ac.uk

Abstract: Effective communication is crucial for ensuring inclusivity, yet the hard-of-hearing community faces significant barriers due to a shortage of qualified interpreters. British Sign Language (BSL), officially recognised in the UK, relies on hand gestures, facial expressions, and body movements. However, limited interpreter availability necessitates technological solutions to bridge the communication gap between signers and non-signers. This study proposes a real-time, vision-based BSL recognition system using computer vision and deep learning to interpret fingerspelling and six commonly used BSL words. The system employs OpenCV for video capture, MediaPipe for hand feature extraction, and Long Short-Term Memory (LSTM) networks for sign classification. A dataset incorporating left- and right-handed signers achieved a 94.23% accuracy rate for 26 fingerspelling gestures and 99.07% for six words. To enhance usability, a graphical user interface was developed, enabling seamless real-time interaction. These findings demonstrate the potential of AI-driven sign language recognition to improve accessibility and foster more inclusive communication for the hard-of-hearing community.

Keywords: British Sign Language; Realtime Recognition; Media Pipe; Hard-of-Hearing; LSTM; OpenCV

1. Introduction

Sign language is a vital mode of communication, particularly where verbal interaction is not feasible. It is widely used across various settings, such as in sports, traffic control, and by individuals with speech or hearing impairments. For the deaf-mute community, sign language offers a robust means of expression, combining hand signs, lip movements, and facial expressions to convey meaning. Historically, the development of sign language began in the 17th century with Juan Pablo de Bonet's instructional work for the deaf [1], and it gained prominence through pioneers like Abbe Charles Michel de L'Epee, who established the first public school for the deaf in 1771.

Despite its widespread use, sign language varies across regions, with over 200 distinct versions globally, including American Sign Language (ASL) and British Sign Language (BSL), which differ significantly in finger-spelling techniques. BSL, used by 151,000 people in the UK, was officially recognised as a minority language in 2003, and recent legislative efforts further advocate for its formal recognition in England, Wales, and Scotland. However, the scarcity of trained interpreters – only 1557 registered in the UK – highlights a gap in accessibility for BSL users, necessitating innovative technological solutions [2]. While language translation apps have revolutionised communication, sign language recognition remains largely underdeveloped. This research seeks to explore how Artificial Intelligence (AI) and machine learning can be applied to develop a real-time visual recognition system for British Sign Language (BSL), addressing the communication barriers faced by the deaf community and offering an afford-

able, scalable solution [3, 4].

1.1. Traditional Methods of British Sign Language (BSL) Recognition

The real-time recognition of British Sign Language (BSL) poses significant challenges due to the intricate nature of sign gestures and their variations. Traditional approaches to Sign Language Recognition (SLR) often emphasise static image-based methods, limiting their applicability to dynamic real-time contexts. Several studies on real-time vision-based sign recognition offer insights into different techniques used across sign language systems. For example, Patel and Patel [5] implemented a real-time Indian Sign Language recognition system with a 98.7% success rate using a Support Vector Machine (SVM) classifier. Their system employed skin-color detection and Histogram of Oriented Gradients (HOG) for feature extraction. However, this method was tested only with static images and faced performance issues due to its reliance on skin-color detection, which can be impacted by variations such as rolled-up sleeves.

Similarly, Tanvir et al. [6] achieved 99.72% accuracy in Bangla Sign Language recognition using a 2D Convolutional Neural Network (CNN). While successful, this approach also dealt exclusively with static signs, limiting its real-world applicability to sequences of hand movements. Despite high recognition rates, both systems are constrained by their focus on static images rather than continuous gestures. A notable contribution to BSL recognition was presented by Liwicki and Everingham [7], who developed an automatic recognition system for finger-spelled BSL words using a dataset of 1,000 videos. Their approach integrated HOG descriptors and a Hidden Markov Model (HMM) for hand segmentation and classification, achieving a 98.9% accuracy rate for 100 forenames. However, their model struggled with larger lexicons and similar-spelling words, indicating difficulties in recognising nuanced hand movements. Additionally, both hands were treated as a single shape descriptor, limiting the system's ability to distinguish palm-based letters such as L, M, and N in various orientations [8].

In contrast, Kumar [9] proposed a deep learning framework to improve BSL recognition using CNN, median filtering, and edge detection techniques. The system achieved 98% accuracy for BSL fingerspelling but was similarly limited to pre-recorded videos rather than real-time applications. Real-time testing and adaptation for both left- and right-handed users were also absent from this study, which further limits its practical deployment. Several works have focused on static image recognition but encountered issues with sequence processing. For instance, Rambhau [10] implemented a real-time two-hand gesture system for BSL recognition using background subtraction and skin segmentation techniques. Yet, this approach was confined to recognising static alphabet gestures and did not detail performance metrics, making it difficult to assess its real-time capabilities.

More recently, Olszewska and Quinn [4] developed an intelligent BSL recognition system embedded in a smartphone. This system utilised a self-created dataset of 2,600 images, applying HOG and an SVM classifier to achieve 99% accuracy for fingerspelling. However, the dataset did not cover left- and right-handed signs, and the system was not tested on dynamic sequences, further limiting its real-time application. Other approaches, such as those by Bird et al. [11], incorporated CNNs like VGG16 and Multi-Layer Perceptron (MLP) for feature extraction in a multimodal system, achieving 94.44% training accuracy. However, the model's accuracy on unseen data dropped to 76.5%, reflecting the difficulties in recognising gestures that are not present in the training set. Similarly, Buckley et al. [12] developed a CNN-based system for predicting 19 static BSL signs, with real-time evaluation achieving an average accuracy of 89%. While this system employed OpenCV for real-time testing, it still focused primarily on static signs, limiting its robustness for continuous gesture recognition. A novel method by Hameed et al. [13] incorporated radar sensors and deep learning models like InceptionV3 and VGG16 to preserve user privacy during BSL recognition. This system achieved a 93.33% accuracy for six emotion-related signs but required the user to stand at a specific distance from the radar, limiting its practical real-time use [14].

Across these works, a common limitation is the focus on static image recognition without accounting for the fluid nature of sign language, where hand movements often occur in sequences. Moreover, many studies do not consider both left- and right-handed signers, introducing potential biases in recognition systems. Thus, there is a growing need for more advanced methodologies capable of handling dynamic, real-time sign language recognition while accommodating the variability of individual signers.

1.2. AI and Real-Time Recognition System

Artificial intelligence (AI) has revolutionised problem-solving and decision-making by enabling computers to mimic human cognition. As IBM Cloud Education [15] explains, AI's key subsets – machine learning and deep learning – are critical in addressing complex challenges. Deep learning, in particular, is adept at managing unstructured datasets like audio or video, using neural networks to solve both supervised and unsupervised problems. Computer vision, a branch of AI, focuses on developing systems that derive meaningful information from visual data, learning patterns to make predictions [15].

Sign Language Recognition (SLR) is an ongoing research area leveraging AI and machine learning techniques to bridge communication gaps for the deaf community. SLR systems follow a multi-stage process: data acquisition, pre-processing, region-of-interest detection, feature extraction, classification, and real-time prediction. The first phase, data acquisition, utilises either vision-based or sensor-based methods [16]. Vision-based methods apply computer vision and image processing techniques, but face challenges such as noise from lighting conditions, background interference, and low detection accuracy, especially when hand movements overlap.

In contrast, sensor-based methods like data gloves - equipped with sensors and trackers - offer high reliability but are inconvenient for users, while electromyography (EMG) measures muscle signals but introduces complex noise during feature extraction [16]. Feature extraction, the next critical step, plays a key role in system accuracy. Methods like edge detection (e.g., canny edge detector) or skin detection, combined with hand motion, help isolate the hands from the background. Sensor-based systems, such as data gloves and accelerometers, reduce feature extraction complexity but impede natural interaction. Techniques like Principal Component Analysis (PCA) simplify datasets, while Scale Invariant Feature Transform (SIFT) algorithms handle variations in rotation and scaling. Additionally, tools like Microsoft Kinect and Leap Motion controllers provide robust 3D tracking, enhancing real-time SLR performance.

1.2.1. Sign Language Recognition Using Media Pipe

The Recent advancements in computer vision have significantly enhanced sign language recognition systems, moving from traditional techniques such as background subtraction and hand detection to more sophisticated frameworks like MediaPipe. Launched by Google in 2019 [17], MediaPipe provides pre-trained machine learning models for hand, face, and pose detection, offering a more streamlined and accessible tool for researchers and developers in sign language detection [18]. Despite its relatively recent introduction, MediaPipe has been effectively utilised in a range of studies, although its application in sign language recognition remains limited. For instance, Halder and Tayade [18] used MediaPipe in combination with a Support Vector Machine (SVM) to recognise signs from American, Indian, Italian, and Turkish Sign Languages. Their approach, which was trained on static images of single-handed signs, achieved an impressive average accuracy of 99%. However, the model was limited to recognising static signs only, potentially restricting its broader applicability in dynamic sign language interpretation.

In a different approach, Duy Khuat et al. [19] integrated MediaPipe with Long Short-Term Memory (LSTM) networks to detect multi-hand Vietnamese Sign Language. While the dataset was self-constructed, the study encountered challenges such as overfitting due to invalid data frames, resulting in a relatively low accuracy of 63%. This highlights the importance of rigorous data cleaning and preprocessing in achieving reliable results. Alvin et al. [20] employed MediaPipe alongside K-Nearest Neighbours (KNN) to classify American Sign Language (ASL). Their model achieved a 94% accuracy but excluded dynamic signs like 'J' and 'Z' due to KNN's inability to handle the z-coordinate of hand landmarks, thereby limiting its effectiveness for comprehensive sign language detection. Similarly, Bora et al. [21] utilised Mediapipe to develop a visual solution for the sign language recognition problem in Assamese language, regional Indian language, where 2094 data points was generated. The model's focus on single-handed signs, however, suggests a gap in recognising more complex, two-handed gestures that are common in many sign languages.

In a more recent study, Subramanian et al. [22] proposed integrating MediaPipe with an Optimised Gated Recurrent Unit (MOPGRU) to address challenges in dynamic sign language recognition. Their model outperformed other recurrent neural networks such as LSTM and BiLSTM, achieving a 95% accuracy on a dataset of 12 signs. However, the study noted that sequence prediction using LSTM and BiLSTM was less effective due to the limited dataset, underscoring the need for larger, more diverse datasets for training deep learning models. These studies illustrate

that while MediaPipe provides a strong foundation for sign language recognition, challenges remain, particularly in recognising dynamic and multi-handed signs. Building on these insights, future work could focus on integrating MediaPipe with more robust deep learning models, allowing for real-time validation and improved recognition of both single-handed and multi-handed signs for languages such as British Sign Language (BSL).

2. Methods

The Developing British Sign Language (BSL) recognition system integrates computer vision with machine learning techniques for real-time sign detection. A supervised machine learning model is trained on a BSL dataset to recognise hand gesture patterns. Once trained, the model is saved and used for live prediction. The system activates the camera via a graphical user interface (GUI), capturing hand signs frame by frame. Using feature extraction, the machine learning model predicts the sign's label and displays it as text on the screen (see **Figure 1** for model architecture).
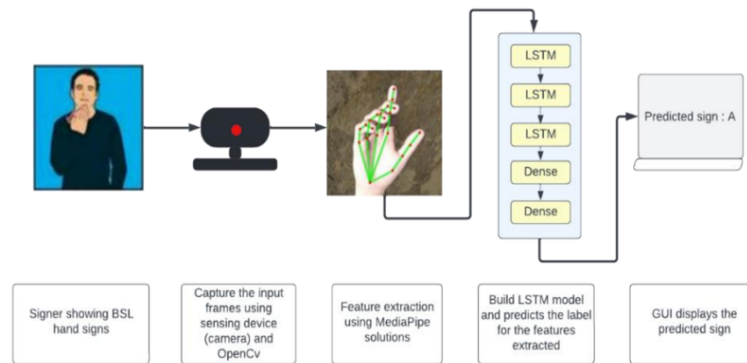

Figure 1. Architecture of British Sign Language (BSL) Recognition System.

The development of the proposed system followed a structured four-stage machine learning process, emphasising the DCSA approach (*Define problem, Collect data*, *Select model*, *Apply model*). Each stage was systematically executed, ensuring alignment with best practices in data science. **Figure 2** provides a visual summary of the stages, and the subsequent sections offer a detailed breakdown of each step.
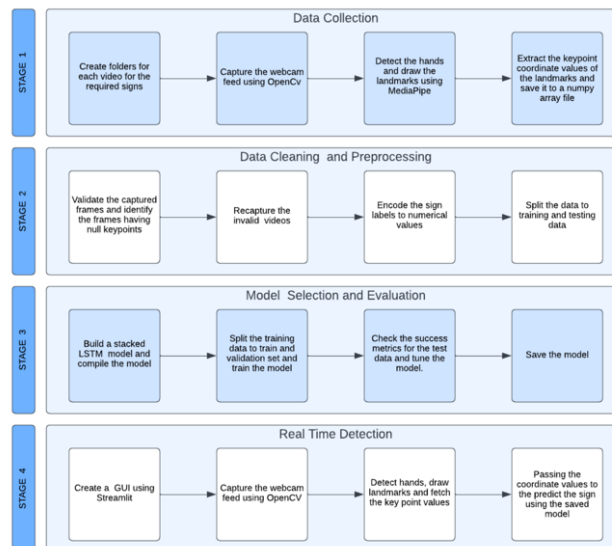

Figure 2. Stages of BSL recognition system development.

The system configuration for processing activities is critical to ensuring efficient data collection, cleansing, preprocessing, and modelling. Each of these steps is crucial for building a robust data pipeline capable of supporting accurate and scalable system outcomes.

2.1. System Configuration (Hardware and Software Configuration)

The hardware and software configurations employed for this implementation are adopted to optimise performance and facilitate efficient execution of the computer vision and machine learning tasks.

Hardware

The system utilised an HP Wide Vision 720p HD webcam for capturing video input. The computing environment consisted of a Windows 11 machine with an Intel Core i5-1155G7 processor (up to 4.5 GHz, 4 cores, 8 threads) and 8 GB DDR4-3200 MHz RAM. The integrated Intel Iris Xe Graphics GPU provided additional computational support for image processing tasks.

Software

Python 3.8 served as the primary interpreter, with a range of specialised libraries to support application development:

- **Computer Vision**: OpenCV-python (v4.5.5.64) and MediaPipe (v0.8.10) were used for real-time image processing.
- **Machine Learning**: The scikit-learn (v1.0.2), Keras (v2.8.0), and TensorFlow (v2.8.0) libraries facilitated model development and training.
- **Visualization**: Matplotlib (v3.5.1) was employed for generating visual insights from data.
- **GUI**: Streamlit (v1.8.1) enabled the creation of an intuitive user interface.
- **Additional Libraries**: Pyttsx3 (v2.90) for text-to-speech, along with Pandas (v1.4.0) and NumPy (v1.22.2) for data manipulation and numerical computing.

2.2. Data Collection for Real-Time Sign Language Recognition

Real-time sign language recognition remains a challenging area of research due to the scarcity of publicly available datasets, especially for British Sign Language (BSL). Unlike American Sign Language (ASL), BSL datasets are often limited or non-existent in open repositories. Consequently, researchers frequently rely on custom datasets created specifically for their studies.

2.2.1. Data Collection Methods

To address the lack of readily available BSL datasets, data collection for this study involved capturing videos of signs using two primary methods: sourcing videos images from Video platforms or self-recording. Self-recording, however, was chosen for its advantages in privacy and control over data quality. A pose captures the body's position at a specific moment using a set of skeletal landmarks, each representing key body parts like the shoulders and hips. The spatial relationships between these landmarks define and differentiate various poses [23]. Using a webcam, we captured signs in real-time to minimise data storage needs and streamline feature extraction. To optimise data storage and processing, we utilised a method where key data points were extracted directly from video frames rather than storing the entire video. This approach was facilitated by the MediaPipe framework, an open-source tool that supports cross-platform development and offers solutions for hand, face, and pose detection [24]. Specifically, MediaPipe's holistic model provides 21 3D landmarks per hand, which are essential for accurate sign language recognition as shown in **Figure 3**. The landmarks are represented by coordinates (x, y, z), with x and y normalised to the image dimensions and z indicating depth [24].
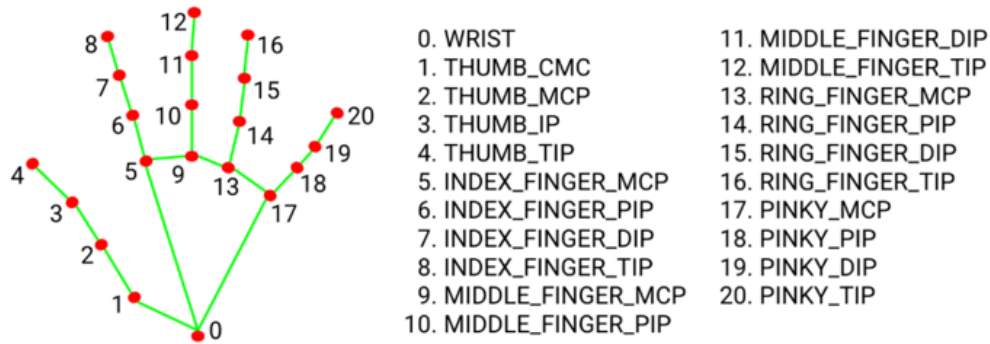
Figure 3. MediaPipe Hand Landmarks [24].

OpenCV, a widely used computer vision library, was employed for real-time video processing. The library's functions, such as 'VideoCapture' for accessing webcam feeds and 'imread' for image processing, facilitated efficient data collection and validation. The integration of OpenCV with MediaPipe allowed us to preprocess images effectively by performing tasks such as noise reduction and skin colour detection [25].

2.2.2. Data Collection Protocol

For fingerspelling, a dataset consisting of 120 video sequences (60 for each hand) was created. Each video captured 30 frames per second at a resolution of 640 × 480 pixels. Similarly, 180 videos were recorded for BSL words, with 90 videos per hand (as shown in **Table 1**). This setup enabled the collection of comprehensive data for both hand signs and individual BSL words.

Table 1. Summary of Collected Data.

Category	Classes	Number of Videos per Class		Frames per Video
		Right-handed Signer	Left-handed Signer	
Fingerspelling	26	60	60	30
BSL Words Signs	6	90	90	30

The signs were referenced from **Figure 4** "British Sign Language": Fingerspelling [26] and verified against multiple sources to ensure accuracy. The collected data was organised in a hierarchical directory structure, where each sign had its own folder, and videos were stored in subfolders.

(a)

Figure 4. *Cont.*

(**b**)
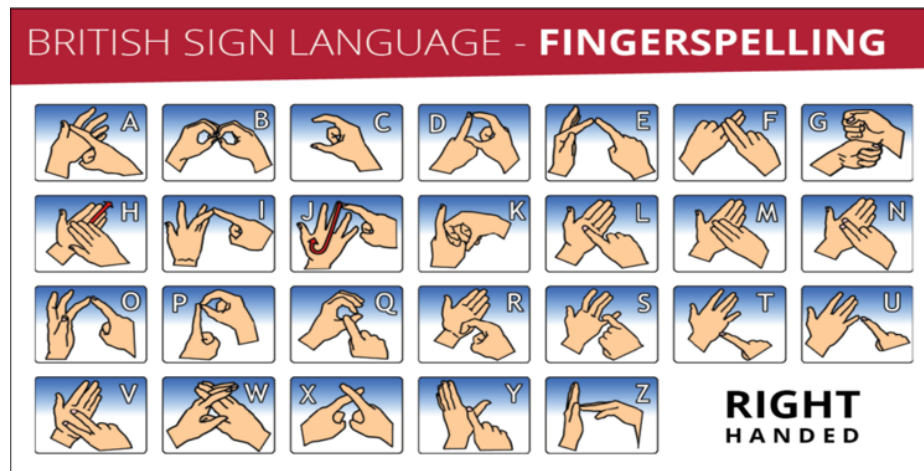
Figure 4. (**a**) BSL Fingerspelling Alphabet Charts [26]; (**b**) BSL Words Chart [26].

2.2.3. Data Collection Algorithm

An automation script in Python was developed to streamline data collection:

Step 1. Initialise MediaPipe and OpenCV:
Set up the MediaPipe holistic solution and OpenCV for real-time image capture.
Step 2. Process Frames:
Convert frames from BGR to RGB, process them with MediaPipe, extract key points.
Step 3. Draw Landmarks:
Visualise detected landmarks on the frames.
Step 4. Extract Key Points:
Retrieve and store the (x, y, z) coordinates for each hand.
Step 5. Organise Data:
Create directories and save the key points as NumPy arrays.

Voice commands were used to guide the signer through the process, ensuring smooth transitions between signs. The Python script included functionality for displaying reference images, issuing voice commands, and managing delays to assist the signer. Each frame contained 126 key points, encompassing both hands' landmarks. The dataset will be utilised for training and testing machine learning models, comparing performance on both a small scale (fingerspelling) and a larger scale (BSL words). This structured approach ensures that the data collection is robust and suitable for future extensions, potentially including BSL sentences.

2.3. Data Cleaning and Preprocessing

Effective data cleansing is a critical step in the machine learning process, ensuring model accuracy and reliability. The quality of data directly influences model performance and predictive outcomes [27]. In this case, hand detection failures, such as blurred frames or inconsistent lighting, result in keypoint extraction errors where landmark values are set to zero, rendering those frames invalid for analysis.

2.3.1. Data Cleansing

To address this, a Python algorithm was implemented to systematically identify and remove invalid frames. The algorithm iterates through each sign and its corresponding video frames, detecting frames where no hand

landmarks are captured. These invalid videos are flagged for recapture and revalidation to ensure data accuracy. For instance, three videos representing the sign for "C" failed to capture hand landmarks. After recapture and reprocessing, these frames were validated, improving the dataset's integrity. This process guarantees that only accurate data is fed into the model for training, significantly enhancing performance.

2.3.2. Data Preprocessing

Data preprocessing is essential for preparing raw data for machine learning models. In this study, the x and y coordinate values obtained from hand landmarks were normalized to the range [0,1], ensuring that variations in hand size did not affect real-time detection accuracy. Machine learning algorithms require numerical inputs. Therefore, categorical class labels (alphabets) were converted into numerical values using the 'enumerate' function, which created a dictionary of key-value pairs representing each class. All video instances, totalling 3120 (120 instances per 26 classes), were stored in arrays for training. Each video had 30 frames, each with 126 features. The corresponding labels were one-hot encoded using Keras' "to_categorical" function, transforming each label into a binary vector. This step was crucial as the model used categorical cross-entropy loss, which requires one-hot encoded labels for training.

2.3.3. Data Split for Model Training

The dataset was divided into 80% training data and 20% test data to evaluate model performance on unseen data. Additionally, 20% of the training data was set aside for validation during model training. The final structure of the dependent variable (y) consisted of a matrix with binary values [0, 1], where rows represented video instances and columns corresponded to the class labels. The train-test split resulted in 2496 videos for training and 624 for testing, as shown in **Table 2**.

This preprocessing approach ensured that the model could handle the complexity of multi-class classification efficiently, preparing the data for optimal performance during training and evaluation [28, 29].

Table 2. Data split (Train, Test & Validation).

Available Data	Features/Independent Variable (X)	Labels/Dependent Variable (y)
Total Data	(3120, 30, 126)	(3120, 26)
Train Data	(1997, 30, 126)	(1997, 26)
Validation Data	(499,30, 126)	(499,26)
Test Data	(624, 30, 126)	(624, 26)

2.4. Model Architecture and Implementation

The model implementation covers the selection of models, architecture, advantages, and evaluation methods used for British Sign Language (BSL) fingerspelling and word classification. Two models - Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) - were chosen based on their performance in similar contexts. Yang et al. [30] compared the performance of LSTM and GRU neural networks on varying dataset sizes in the Yelp dataset, showing that GRU's simpler structure reduces computational time without sacrificing performance, particularly for small datasets. This makes both models suitable candidates for evaluating performance on the newly created BSL dataset.

The implemented sequential model consists of three LSTM layers stacked with three dense layers, utilising the ReLU activation function for the LSTM layers. ReLU (Rectified Linear Unit) is efficient as it does not activate all neurons simultaneously, leading to faster computation [31]. For the final dense layer, a softmax function is applied, which is appropriate for multiclass classification. As Sharma et al. [31] state, softmax outputs probabilities between 0 and 1, representing the likelihood of each class. The model was compiled using the Adam optimizer, with a default learning rate of 0.001. Adam is widely used due to its memory efficiency and effectiveness in handling large datasets or parameters [32]. Categorical cross-entropy was employed as the loss function, given the one-hot encoded labels, and categorical accuracy was used as the evaluation metric. Training was conducted over 200 epochs with early stopping set to monitor validation loss, avoiding overfitting. Early stopping patience was set to 15 epochs, following Vijay [32], who recommends a patience threshold of 10% of the total epochs.

2.5. Graphical Interface for Realtime Detection

To evaluate model performance, the best-performing model was selected based on predefined success metrics. This model underwent rigorous testing in a real-time environment, involving five users. The testing process was conducted using the PyCharm IDE to ensure reliability and accuracy in predictions. To enhance user interaction, the critical analysis code was integrated into a graphical user interface (GUI) using the Streamlit framework, a user-friendly, open-source tool that simplifies web application development with Python scripts. Streamlit was chosen for its ability to expedite development time and facilitate the creation of multi-page applications with intuitive widgets. Despite its advantages, one challenge encountered was the relatively slow direct live streaming in Streamlit compared to OpenCV's performance. To address this, the application was designed to allow users to activate their camera through OpenCV via button widgets within the Streamlit interface, balancing ease of use with efficient performance.

3. Results

Provide a concise and precise description of the experimental results, their interpretation as well as the experimental conclusions that can be drawn.

3.1. Model Performance Analysis: LSTM vs. GRU

Accuracy and loss plots are vital tools for assessing the performance of machine learning models, particularly for identifying issues such as underfitting, overfitting, or optimal fitting. For both LSTM and GRU models, accuracy and loss plots in **Figure 5a,b** were generated using Matplotlib to visually evaluate these aspects throughout the training process.
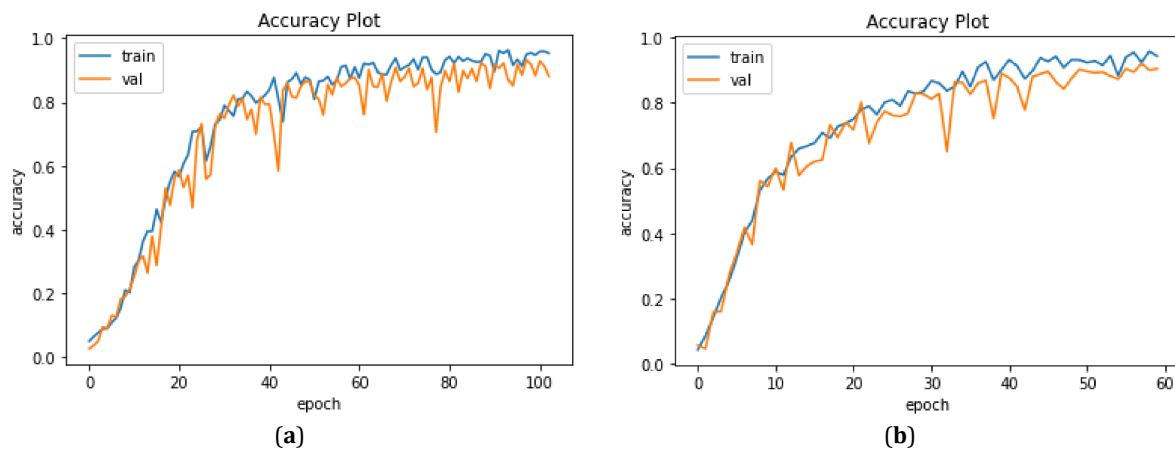

(a)   (b)

Figure 5. (**a**) LSTM Accuracy Plot, Accuracy plot of LSTM Model; (**b**) GRU Accuracy Plot, Accuracy plot of GRU Model.

The accuracy plots depict the progression of the 'categorical accuracy' metric for both training and validation datasets over each epoch. Similarly, loss plots represent the 'categorical_crossentropy' values at each epoch, providing insights into the model's convergence behaviour. Early stopping, a regularization technique, was employed to monitor validation loss, halting training when no improvement was observed or when validation loss began to increase. This approach prevents overfitting by ensuring the model does not continue to train unnecessarily.

For the GRU model, training ceased at the 60th epoch, with the weights being restored from the epoch that yielded the best validation performance. This process resulted in a final training accuracy of 94.24% and a validation accuracy of 90.4%. When evaluated on an unseen test dataset, the GRU model achieved a test accuracy of 91.5%. As depicted in **Figure 2**, the GRU model's training accuracy showed a significant increase from 57% at the 10th epoch to 94% by the 60th epoch.

The LSTM model, on the other hand, continued training until the 103rd epoch, reaching a training accuracy of 95.29% and a validation accuracy of 88.20%. The model's performance on the test dataset resulted in an accuracy of 94.23%. **Figure 6b** illustrates the LSTM model's accuracy growth, with training accuracy improving from 20.24% at the 10th epoch to 95.29% by the 103rd epoch. Additionally, the loss plot in **Figure 6a** shows a decrease in validation loss from 2.3666 at the 10th epoch to 0.3832 at the 103rd epoch, indicating effective training and convergence of the model.
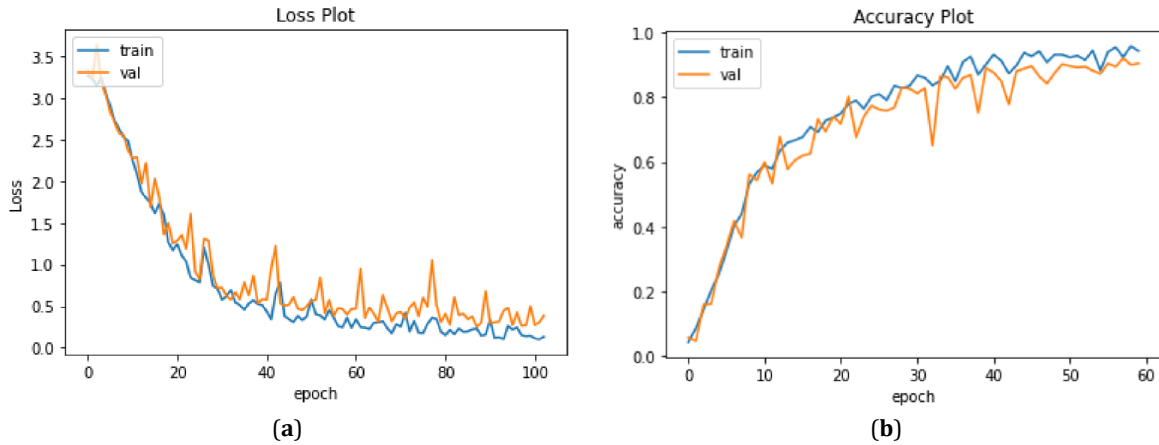

(**a**)   (**b**)

Figure 6. (**a**) LSTM Loss Plot, Loss plot of LSTM Model; (**b**) GRU Loss Plot, Loss plot of GRU Model.

The results in **Table 3** underscore the effectiveness of both LSTM and GRU models, with LSTM slightly outperforming GRU in test accuracy, albeit with a longer training period.

Table 3. Summary of accuracy obtained for train, validation, and test data for LSTM and GRU models.

Model	Train	Validation	Test
LSTM	95.25	88.2	94.23
GRU	94.24	90.4	91.5

3.2. Analysis of Model Performance Using Normalized Confusion Matrices

To evaluate the performance of our models, we employed a 26 × 26 normalized confusion matrix for both the LSTM and GRU models, comparing predicted versus true values across the 26 alphabet classes. This approach was necessary due to the imbalanced nature of the test data, where the number of instances varies across classes. Normalizing the confusion matrix allows for a more accurate assessment of prediction accuracy across all classes, mitigating the effects of class imbalance [33].

In the normalized confusion matrix for the LSTM model, nine classes achieved perfect prediction accuracy (True Positive rate = 1.0). Additionally, twelve classes had accuracy rates between 0.9 and 1.0, four classes between 0.8 and 0.9, and one class with an accuracy of 0.72. Conversely, the GRU model's normalized confusion matrix indicates seven classes with a perfect True Positive rate, nine classes with accuracy between 0.9 and 1.0, eight classes between 0.8 and 0.9, and two classes with accuracies of 0.73 and 0.68, respectively (**Figure 7**). These results highlight the strengths and weaknesses of both models in handling class predictions, particularly under conditions of class imbalance. The normalized confusion matrix provides critical insights into the models' reliability across different classes, offering a comprehensive view of performance beyond simple accuracy metrics.
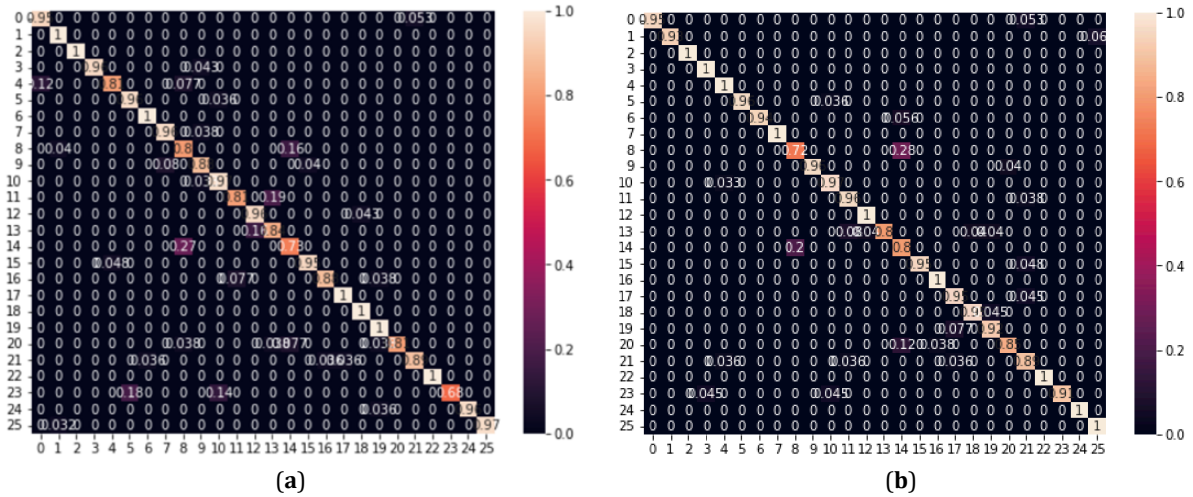
Figure 7. (**a**) GRU Matrix, Normalized confusion matrix for GRU; (**b**) LSTM Matrix, Normalized confusion matrix for LSTM.

3.3. F1 Score: A Critical Metric for Model Evaluation

While accuracy and true positives are often highlighted in model performance evaluation, they are insufficient on their own to provide a complete picture. False negatives and false positives play a crucial role in understanding a model's weaknesses, specifically its tendency to miss classifications or make incorrect ones. Recall and precision are essential metrics for this purpose, with recall focusing on false negatives and precision on false positives. However, for a more comprehensive assessment - especially when dealing with imbalanced datasets where both types of errors are significant - the F1 score is pivotal.

In the evaluation of the LSTM model, F1 scores for each class are presented in **Figure 8a**. The scores range from 0 to 1, with a score of 1 indicating optimal performance. A threshold of 0.9 was set for this analysis. Four classes ('I', 'N', 'O', 'V') fall below this threshold, with the class 'O' having the lowest F1 score at 0.63. Similarly, the GRU model's performance is depicted in **Figure 8b**, where eight classes ('E', 'I', 'J', 'L', 'N', 'O', 'U', 'X') fall below the 0.9 threshold, with 'O' again recording the lowest score at 0.69. These results underscore the importance of using the F1 score in performance evaluation, particularly in cases of class imbalance, to ensure a more accurate and fair assessment of the model's predictive capabilities. Before deploying the Long Short-Term Memory (LSTM) model to the graphical user interface (GUI), a real-time evaluation was conducted to assess its predictive accuracy during live usage. This process involved testing with five participants, each using a webcam to fingerspell various signs. The system captured frames from the video feed, extracted relevant features, and utilised the pre-trained model to predict the corresponding sign labels. The validation process required the participants to manually verify the accuracy of the model's predictions. If the predicted sign matched the intended one, participants logged a "True" (T) value; otherwise, they recorded a "False" (F). These results were systematically stored and analysed, with the outcome data being tabulated for both right- and left-hand fingerspelling signs, as detailed in **Appendix A**. Following this evaluation, the model was integrated into the web application's GUI, making it accessible for user interaction. This real-time validation was crucial in ensuring the model's robustness and accuracy in a live environment before public deployment.
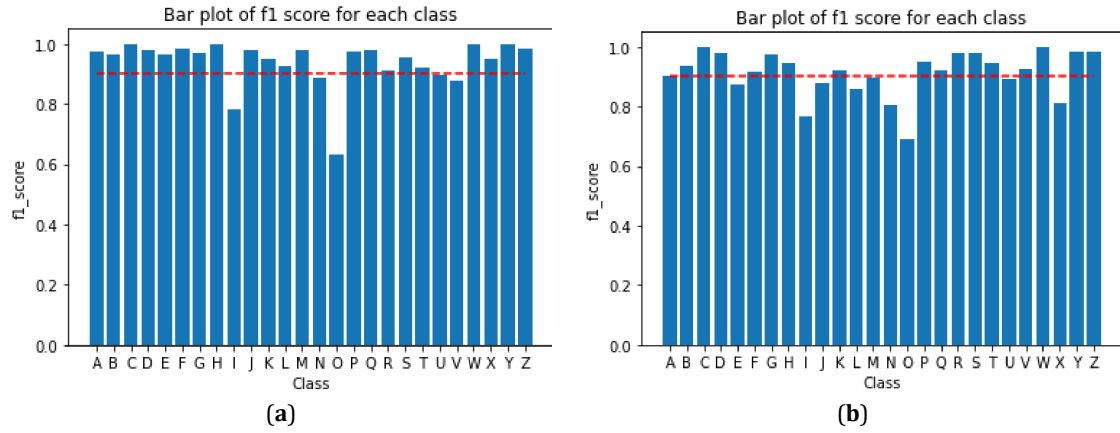
Figure 8. (**a**) LSTM F1 Score, Fingerspelling Bar plot of F1 score for LSTM; (**b**) GRU F1 Score, Fingerspelling Bar plot of F1 score for GRU.

4. Discussion

The evaluation of LSTM and GRU models revealed that both architectures performed effectively in addressing the research problem, though with notable differences in accuracy and other key metrics. The LSTM model achieved a higher test accuracy of 94.23% compared to the GRU's 91.5%, indicating a slight edge for LSTM in overall performance.

A deeper analysis using normalized confusion matrices provided insights into the true positive rates across different classes. The LSTM model successfully predicted more than 80% of classes accurately, with the exception of class 8 ('I'), which had a 72% accuracy rate. The GRU model, on the other hand, struggled more significantly with certain classes, such as 'X' (class 23) and 'O' (class 14), achieving accuracy rates of 68% and 73%, respectively. This suggests that while both models are generally effective, LSTM demonstrated more consistent performance across the dataset. Further examination of the multilabel confusion matrix highlighted the models' tendencies toward misclassification. LSTM exhibited fewer false negatives (FN) and false positives (FP) for critical classes such as 'I', 'N', 'O', and 'V'. For instance, class 'I' under LSTM had 3 FPs and 7 FNs, whereas GRU recorded 7 FPs and 5 FNs. This indicates that LSTM had a lower rate of misclassification, contributing to its higher overall reliability.

Comparing the F1 scores of both models, LSTM outperformed GRU in 22 out of 26 classes, with a higher overall F1 score (0.94 vs. GRU's 0.91). Despite GRU showing slightly better results in a few specific classes, the LSTM model's superior performance in the majority of cases, coupled with a more balanced classification, justifies its selection for real-time validation.

In terms of computational efficiency, GRU's fewer trainable parameters (73,258) compared to LSTM's 96,362 suggested faster training times for GRU. However, the computational cost remained manageable for the dataset used, and the LSTM's slight increase in complexity did not hinder its performance, making it a suitable choice given the problem's requirement. Interestingly, when tested on a smaller dataset of 6 BSL words with 180 videos per class, the LSTM model achieved near-perfect accuracy (99.5%) and an F1 score of 1. This outcome underscores the potential for further improving LSTM's performance with larger datasets, particularly in the context of BSL fingerspelling. Overall, the LSTM model was selected as the preferred model due to its superior accuracy, lower misclassification rates, and better generalisation across classes, making it more robust for real-time applications.

Table 4 presents a comparative analysis of our LSTM-based sign language recognition model against previous works utilising MediaPipe. Our model achieved a notable accuracy of 94.23% across 26 classes, surpassing the performances of Adhikary et al. [34] with 11 classes, Subramanian et al. [22] with 12 classes, and Alvin et al. [20] with 24 static classes. Although Halder and Tayade [18] reported higher accuracy, their dataset was limited to single-hand signs, indicating a narrower scope and less complexity compared to our approach.

Table 4. Comparative Analysis of MediaPipe-Based Sign Language Recognition.

Author, Year	Description	Performance
Halder and Tayade [18]	MediaPipe along with Support Vector Machine for static images: American signs (26 alphabets and 10 numbers), Indian (24 alphabets), Italian (22 alphabets), and Turkey (10 numbers)	99%
Duy Khuat et al. [19]	MediPipe with LSTM is used to detect Vietnamese Sign Language for 15 words	63%
Alvin et al. [20]	MediaPipe with KNN to detect 24 static American Sign Language	94%
Adhikary et al. [34]	MediaPipe and Random Forest Classifier for 11 Classes	97.4%
Subramanian et al. [22]	MediaPipe with an optimized Gated Recurrent Unit (MOPGRU) for 12 signs	95%
BSL Intelligent Recognition System (Current work)	MediaPipe with LSTM for BSL Alphabets (26 letter)	94.23%
	MediaPipe with LSTM for BSL words (6 words)	99.0%

5. Conclusions

This Sign language recognition remains a dynamic field, driven by advances in artificial intelligence and machine learning. Recent developments have centered on leveraging computer vision and deep learning techniques to create cost-effective, vision-based systems that facilitate real-time recognition across various sign languages. Among these techniques, Convolutional Neural Networks (CNNs) have emerged as the leading algorithm for image classification. However, for sequential data such as video or continuous frames, Recurrent Neural Networks (RNNs) like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are preferred due to their ability to retain and process essential information over time.

This study adopted cutting-edge methods for data collection and feature extraction specifically tailored for British Sign Language (BSL) fingerspelling and selected signs. Data was captured in real-time using OpenCV, with MediaPipe employed for efficient feature extraction. This approach allowed for the handling of large datasets by converting frame data into NumPy arrays containing hand feature coordinates, significantly simplifying the preprocessing pipeline typically required in image classification tasks.

The LSTM and GRU models were selected for their proficiency in managing sequential data, with LSTM outperforming GRU, achieving an F1 score of 0.94 and a test accuracy of 94.23%. The system was further evaluated in real-time scenarios, where it demonstrated an 86% accuracy rate across tests with five different signers. The integration of a Streamlit web application provided a user-friendly interface, enhancing accessibility and usability.

While the proposed methods successfully achieved the study's objectives, certain limitations were identified. For instance, the MediaPipe-based approach exhibited flickering in landmark detection during real-time application. Additionally, the dataset faced orientation challenges, particularly with palm signs such as 'L', 'M', and 'N' being trained with the non-dominant palm's dorsal side facing the camera, while 'R' and 'V' were trained with the palmar side facing the camera. The current scope is limited to hand movements for fingerspelling and specific words, unaffected by cluttered backgrounds but challenged by the presence of multiple people or hands on screen.

Future work will aim to expand beyond static and dynamic hand sign detection to include continuous sign language recognition, integrating body posture and facial expressions to accommodate a larger vocabulary. This extension will be crucial in developing a fully functional, real-time BSL recognition system.

Author Contributions

Conceptualisation, O.I. and S.S.; methodology, O.I.; software, S.S.; validation, O.I. and S.S.; formal analysis, S.S.; investigation, O.I. and S.S.; resources, S.S.; data curation, S.S.; writing—original draft preparation, O.I.; writing—review and editing, O.I.; visualisation, S.S.; supervision, O.I.; project administration, O.I. and S.S.

Funding

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Informed consent was obtained from all subjects involved in the study.

Data Availability Statement

The dataset used in this study is detailed in the Methods section. Because access to the hosting repository requires user authorization, the raw data cannot be openly shared. The dataset is available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare no conflict of interest.

Appendix A

Table A1. F1 Score, Recall and precision rate for spell fingers of LSTM vs. GRU.

Class	Label	LSTM			GRU		
		F1_score	Recall	Precision	F1_score	Recall	Precision
0	A	0.97	0.95	1.00	0.90	0.95	0.86
1	B	0.97	0.93	1.00	0.94	1.00	0.88
2	C	1.00	1.00	1.00	1.00	1.00	1.00
3	D	0.98	1.00	0.96	0.98	0.96	1.00
4	E	0.96	1.00	0.93	0.88	0.81	0.95
5	F	0.98	0.96	1.00	0.92	0.96	0.87
6	G	0.97	0.94	1.00	0.97	1.00	0.95
7	H	1.00	1.00	1.00	0.94	0.96	0.93
8	I	**0.78**	0.72	0.86	**0.77**	0.80	0.74
9	J	0.98	0.96	1.00	0.88	0.88	0.88
10	K	0.95	0.97	0.94	0.92	0.97	0.88
11	L	0.93	0.96	0.89	0.86	0.81	0.91
12	M	0.98	1.00	0.96	0.90	0.96	0.85
13	N	0.89	0.80	1.00	0.81	0.84	0.78
14	O	0.63	0.80	0.52	0.69	0.73	0.65
15	P	0.98	0.95	1.00	0.95	0.95	0.95
16	Q	0.98	1.00	0.96	0.92	0.88	0.96
17	R	0.91	0.95	0.88	0.98	1.00	0.96
18	S	0.95	0.95	0.95	0.98	1.00	0.96
19	T	0.92	0.92	0.92	0.95	1.00	0.90
20	U	0.90	0.85	0.96	0.89	0.81	1.00
21	V	0.88	0.89	0.86	0.93	0.89	0.96
22	W	1.00	1.00	1.00	1.00	1.00	1.00
23	X	0.95	0.91	1.00	0.81	0.68	1.00
24	Y	1.00	1.00	1.00	0.98	0.96	1.00
25	Z	0.98	1.00	0.97	0.98	0.97	1.00
Overall		**0.94**	**0.94**	**0.94**	**0.91**	**0.91**	**0.92**

References

1. Van Cleve, J.V.; Crouch, B.A. *A Place of Their Own: Creating the Deaf Community in America*. Gallaudet University Press: Washington, DC, USA, 1989.
2. NRCPD. Registered Sign Language Interpreters in the UK. Available from: https://www.nrcpd.org.uk/registration-figures (accessed on 4 January 2025).
3. Cui, R.; Liu, H.; Zhang, C. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In *Proceedings of The 2017 IEEE conference on computer vision and pattern recognition*, Honolulu, HI, USA, 21–26 July 2017.

4. Park, H.; Lee, Y.; Ko, J. Enabling real-time sign language translation on mobile platforms with on-board depth cameras. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2021**, *5*, 1–30.
5. Patel, P.; Patel, N. Vision Based Real-time Recognition of Hand Gestures for Indian Sign Language using Histogram of Oriented Gradients Features. *Int. J. Next-Gener Comput.* **2019**, *10*, 92–102.
6. Tanvir, M.; Alam, M. S.; Saha, D. K.; et al. Real-Time Recognition of Bangla Sign Language Characters: A Computer Vision Based Approach Using Convolutional Neural Network. In *Proceedings of The 2021 3rd International Conference on Electrical & Electronic Engineering (ICEEE)*, Rajshahi, Bangladesh, 22–24 December 2021.
7. Liwicki, S.; Everingham, M. Automatic recognition of fingerspelled words in British Sign Language. In *Proceedings of The 2009 IEEE computer society conference on computer vision and pattern recognition workshops*, Miami, FL, USA, 20–25 June 2009.
8. Dardas, N.H.; Georganas, N.D. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Trans. Instrum. Meas.* **2011**, *60*, 3592–3607. [CrossRef]
9. Wadhawan, A.; Kumar, P. Deep learning-based sign language recognition system for static signs. *Neural. Comput. appli.* **2020**, *32*, 7957–7968.
10. Rambhau, P.P. Recognition of two hand gestures of word in British sign language (BSL). *Int. J. Sci. Res. Publi.* **2013**, *3*, 1–5.
11. Bird, J.J.; Ekárt, A.; Faria, D.R. British Sign Language recognition via late fusion of computer vision and Leap Motion with transfer learning to American Sign Language. *Sensors* **2020**, *20*, 5151. [CrossRef]
12. Buckley, N.; Sherrett, L.; Secco, E.L. A CNN sign language recognition system with single & double-handed gestures. In *Proceedings of The 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain, 12–16 July 2021.
13. Hameed, H. Privacy-preserving British Sign Language recognition using deep learning. In Proceedings of the 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Glasgow, Scotland, United Kingdom, 11–15 July 2022.
14. Adão, T.; Oliveira, J.; Shahrabadi, S.; et al. Empowering deaf-hearing communication: Exploring Synergies between predictive and generative AI-based strategies towards (Portuguese) sign language interpretation. *J. Imaging* **2023**, *9*, 235.
15. IBM. What is artificial intelligence (AI)? Available Online: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence (accessed on 4 February 2025).
16. Al-Ahdal, M. E.; Nooritawati, M. T. Review in sign language recognition systems. In Proceedings of The 2012 IEEE Symposium on Computers & Informatics (ISCI), Penang, Malaysia, 18–20 March 2012.
17. Google. MediaPipe: Holistic. Available Online: https://mediapipe.dev (accessed on 10 August 2024).
18. Halder, A.; Tayade, A. Real-time vernacular sign language recognition using mediapipe and machine learning. *Int. J. Res. Publi. Rev.* **2021**, *2*, 9–17.
19. Duy Khuat, B.; Thai Phung, D.; Thi Thu Pham, H.; et al. Vietnamese sign language detection using Mediapipe. In *Proceedings of The 2021 10th International Conference on Software and Computer Applications*, Kuala Lumpur, Malaysia, 23–26 February 2021.
20. Alvin, A.; Shabrina, N. H.; Ryo, A.; Christian, E. Hand gesture detection for sign language using neural network with mediapipe. *Ultima Comput. J. Sist. Komp.* **2021**, *13*, 57–62.
21. Adhikary, A. Real-time sign language recognition using MediaPipe and deep learning. *Procedia Comput. Sci.* **2023**, *218*, 1384–1393.
22. Subramanian, B.; Olimov, B.; Naik, S. M.; et al. An integrated mediapipe-optimized GRU model for Indian sign language recognition. *Sci. Rep.* **2022**, *12*, 11964. [CrossRef]
23. Google. Pose detection. In ML Kit - Google for Developers. Retrieved from: https://developers.google.com/ml-kit/vision/pose-detection (accessed on 4 March 2025).
24. Google AI Edge. Hand Landmarks Detection Guide. Available online: https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker (accessed on 2 August 2025).
25. Bradski, G.; Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed. O'Reilly Media: Publisher location, USA, 2008; pp. 102–105.
26. Fingerspelling Alphabet. Available online: https://www.british-sign.co.uk/fingerspelling-alphabet-charts/ (accessed on 4 July 2022).
27. Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann: Publisher location, USA, 2011; pp. 83–120.
28. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*. MIT Press: Cambridge, MA, USA, 2016; p. 163.

29. Chollet, F. *Deep Learning with Python*, 2nd ed. Manning Publications: Shelter Island, NY, USA, 2021.

30. Yang, S.; Yu, X.; Zhou, Y. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *Proceedings of The 2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, Shanghai, China, 12–14 June 2020.

31. Sharma, S.; Sharma, S.; Athaiya, A. Activation functions in neural networks. *Towards Data Sci.* **2017**, *6*, 310–316.

32. Vijay, K.; Krithiga, P.; Kavirakesh, S.; et al. Early Detection of Diabetic Retinopathy Using Deep Convolutional Neural Network. In *Artificial Intelligence and Machine Learning in Big Data Processing*; Geetha, R., Dao, N.N., Khalid, S., Eds.; Springer Nature: Cham, Switzerland, 2023; pp. 315–327.

33. He, H.; Ma, Y. *Imbalanced Learning: Foundations, Algorithms and Applications*, 1st ed. John Wiley & Sons: Hoboken, NJ, USA, 2013; pp. 43–59.

34. Adhikary, S.; Talukdar, A.K.; Sarma, K.K. A vision-based system for recognition of words used in Indian Sign Language using MediaPipe. In Proceedings of The 2021 sixth international conference on image information processing (ICIIP), Shimla, India, 26–28 November 2021.

Publisher's Note: The views, opinions, and information presented in all publications are the sole responsibility of the respective authors and contributors, and do not necessarily reflect the views of UK Scientific Publishing Limited and/or its editors. UK Scientific Publishing Limited and/or its editors hereby disclaim any liability for any harm or damage to individuals or property arising from the implementation of ideas, methods, instructions, or products mentioned in the content.