electronics

MDPI

Article

Real-Time Mobile Application for Translating Portuguese Sign Language to Text Using Machine Learning

Gonçalo Fonseca [1,†], Gonçalo Marques [1,2,†], Pedro Albuquerque Santos [2,3,†] and Rui Jesus [1,2,*,†]

[1] Lisbon School of Engineering (ISEL), Polytechnic University of Lisbon (IPL), 1959-007 Lisboa, Portugal; gcare.fonseca99@outlook.com (G.F.)
[2] NOVA LINCS, ISEL, 1959-007 Lisboa, Portugal; pedro.albuquerque@esce.ips.pt
[3] School of Business Administration (ESCE), Polytechnic University of Setúbal (IPS), 2914-503 Setúbal, Portugal
* Correspondence: rui.jesus@isel.pt
† These authors contributed equally to this work.

Abstract: Communication barriers between deaf and hearing individuals present significant challenges to social inclusion, highlighting the need for effective technological aids. This study aimed to bridge this gap by developing a mobile system for the real-time translation of Portuguese Sign Language (LGP) alphabet gestures into text, addressing a specific technological void for LGP. The core of the solution is a mobile application integrating two distinct machine learning approaches trained on a custom LGP dataset: firstly, a Convolutional Neural Network (CNN) optimized with TensorFlow Lite for efficient, on-device image classification, enabling offline use; secondly, a method utilizing MediaPipe for hand landmark extraction from the camera feed, with classification performed by a server-side Multilayer Perceptron (MLP). Evaluation tests confirmed that both approaches could recognize LGP alphabet gestures with good accuracy (F1-scores of approximately 76% for the CNN and 77% for the MediaPipe+MLP) and processing speed (1 to 2 s per gesture on high-end devices using the CNN and 3 to 5 s under typical network conditions using MediaPipe+MLP), facilitating efficient real-time translation, though performance trade-offs regarding speed versus accuracy under different conditions were observed. The implementation of this dual-method system provides crucial flexibility, adapting to varying network conditions and device capabilities, and offers a scalable foundation for future expansion to include more complex gestures. This work delivers a practical tool that may contribute to improve communication accessibility and the societal integration of the deaf community in Portugal.

Keywords: Portuguese Sign Language (LGP); accessibility; gesture recognition; gesture translation; mobile application; Multilayer Perceptron (MLP); Convolutional Neural Networks (CNNs)

Academic Editors: Ana Martí-Testón and Juan Ernesto Solanes Galbis

Received: 29 April 2025
Revised: 30 May 2025
Accepted: 5 June 2025
Published: 8 June 2025

Citation: Fonseca, G.; Marques, G.; Albuquerque Santos, P.; Jesus, R. Real-Time Mobile Application for Translating Portuguese Sign Language to Text Using Machine Learning. *Electronics* **2025**, *14*, 2351. https://doi.org/10.3390/electronics14122351

1. Introduction

Effective communication is a cornerstone of social integration, yet significant barriers persist for deaf communities globally due to the prevalence of spoken and written languages. In Portugal, Portuguese Sign Language (Língua Gestual Portuguesa—LGP) serves as the natural and primary mode of communication for many deaf individuals, embodying a rich medium for expressing thoughts, emotions, and cultural identity. Despite its official recognition, a considerable gap exists in accessible technological tools designed to facilitate seamless interaction between LGP users and the hearing community, thereby hindering social inclusion and mutual understanding.

The development of automated systems for real-time sign language recognition holds immense potential to bridge this communication divide. However, creating such systems, particularly for languages like LGP, presents substantial technical challenges. These include ensuring robust gesture recognition under variable environmental conditions (e.g., lighting, camera angles), accommodating the natural variability and speed of human signing, and achieving low-latency responses suitable for fluid, real-time communication, especially on widely accessible mobile platforms. While modern computer vision techniques show promise, existing research and development in automated sign language recognition have predominantly focused on more widely spoken sign languages, such as American Sign Language (ASL), leaving LGP relatively underserved. Furthermore, the recognition and translation of complete LGP phrases, with their intricate visual–spatial grammar and syntax, remain a highly complex task beyond the scope of current practical systems.

This work addresses a specific and critical aspect of this broader challenge: the need for an accessible mobile application tailored for the real-time recognition of LGP alphabet gestures (fingerspelling). These gestures constitute a fundamental and universally understood component within LGP, essential for conveying proper names, technical terms, and words lacking specific signs, thus serving as a vital communication bridge. The primary aim of this study is, therefore, to develop and evaluate a practical mobile tool capable of translating LGP alphabet gestures into text in real time. A key objective is to ensure this tool is both accessible and functional under diverse conditions. To achieve this, we developed and comparatively evaluated two distinct machine learning strategies within the application: one prioritizing rapid, on-device recognition for offline use (an image-based Convolutional Neural Network, CNN), and another leveraging remotely processed hand landmark data for potentially different handling of gesture variability (using MediaPipe and a Multilayer Perceptron, MLP). This dual-approach allows for an assessment of the trade-offs between local versus remote processing and image versus feature inputs for mobile LGP recognition, offering valuable operational flexibility.

The contributions of this research are multifaceted. We present a functional, cross-platform mobile application integrating these two recognition models, validated through quantitative performance metrics and qualitative user feedback. Central to this was the creation of a custom LGP alphabet dataset to address the lack of public resources and to train models designed for diverse real-world variations. This study offers comparative insights into the efficacy of different machine learning approaches for LGP recognition on mobile devices, highlighting performance trade-offs regarding speed, accuracy, and operational conditions. Socially, the developed tool has the potential to enhance communication accessibility for the Portuguese deaf community and serve as an interactive LGP alphabet learning aid. Technologically, it provides a validated system and a scalable foundation for future advancements in assistive communication technology, potentially including expansion to a broader LGP vocabulary or adaptation for other sign languages.

The remainder of this paper is organized as follows: Section 2 reviews the pertinent literature on sign language characteristics, LGP context, and the evolution of sign language recognition techniques. Section 3 details the system architecture, the acquisition and preprocessing of the LGP alphabet dataset, the development of the classification models, the mobile application implementation, and the evaluation methodology. Section 4 presents and discusses the performance evaluation of the machine learning models and the usability assessment of the mobile application. Finally, Section 5 summarizes the findings, contributions, and limitations and outlines directions for future work.

2. Related Work

This section provides essential background on sign languages, focusing specifically on Portuguese Sign Language (LGP) and its characteristics relevant to automated recognition. It then reviews the evolution of technical approaches employed in sign language recognition (SLR), highlighting key machine learning concepts and systems pertinent to the methods used in this study.

2.1. Sign Language Characteristics and LGP Context

Sign languages are the primary means of communication for deaf communities globally, employing a complex interplay of hand gestures, facial expressions, and body movements to convey meaning [1]. Unlike spoken languages, they rely on visual–spatial grammar and syntax, unique to each sign language. Linguistic research on American Sign Language (ASL) established sign languages as possessing structured linguistic rules capable of expressing complex and abstract ideas [2,3].

Portuguese Sign Language (Língua Gestual Portuguesa—LGP), officially recognized in the Portuguese constitution [4], follows distinct grammatical principles compared to spoken Portuguese. It heavily utilizes non-manual markers (e.g., facial expressions, body posture) and elements like classifier hand shapes, spatial agreement, and directionality to modulate meaning [5].

Despite its official status, effective communication between deaf LGP users and the hearing community often requires technological aids or human interpreters. This motivates the development of automated sign language recognition (SLR) systems aiming to translate sign gestures into text or speech in real time. However, the visual complexity and nuances of languages like LGP pose significant challenges to automated recognition.

2.2. Evolution of Sign Language Recognition Techniques

Several projects have investigated the translation of sign language [6–12], using various computer vision techniques, such as Template Matching [7] and strategies based on deep learning [9–12]. These studies are divided into static gestures [9,12,13], gestures without movement and represented by an image, and dynamic gestures [7,8,10,11,14,15], defined by their movement, and captured by video.

These works are categorized by type of recognition: typology [12], each gesture represents a letter; isolated gestures [7–9,13,14], individual signs; and continuous gestures [10,11], real, fluid conversations. In capture, gestures and expressions are obtained by video or depth cameras [14], by means of images [9], prerecorded videos [7,10,11,15], or in real time [8,12–14].

The proposed work falls into the category of isolated gestures, capturing frames in real time to detect static gestures in LGP. Therefore, the following works focus on LGP, and the methodologies used, in general, relate to the techniques used.

Early attempts at automated SLR often relied on specialized hardware. Sensor-equipped gloves, for example, could accurately measure hand configurations and finger movements but suffered from being intrusive, costly, and impractical for widespread daily use [16].

Advancements in computer vision enabled the exploration of camera-based systems, with researchers adopting tools such as depth sensors and motion trackers, like the Microsoft Kinect, for gesture recognition. Ribeiro demonstrated its effectiveness for capturing broad hand movements, but its low resolution and limited ability to detect detailed finger positioning made it less suitable for accurately recognizing static letters in sign languages which rely on fine articulation [17]. Similarly, Oliveira combined a CNN with Kinect input to classify sign language gestures, achieving good results in dynamic gesture recognition

but struggling with fine hand shape details due to the sensor's resolution limitations [18]. These limitations highlighted the need for techniques better suited to capturing intricate hand and finger details directly from standard camera imagery.

The advent of deep learning, particularly Convolutional Neural Networks (CNNs), has significantly advanced vision-based SLR [19]. CNNs excel at automatically extracting hierarchical visual features directly from images, effectively processing the spatial relationships inherent in hand shapes and movements without requiring manual feature engineering [20]. This capability stems from their inherent structure: CNNs identify low-level patterns (e.g., edges and contours) in early layers and progressively learn more abstract representations (e.g., specific finger configurations or full hand shapes) in deeper layers. This makes them well suited for the complexities of gesture classification tasks.

Transfer learning is another widely used approach that builds on the power of CNNs. By leveraging models pre-trained on large-scale image datasets like ImageNet [21], transfer learning allows models to be fine-tuned for specific sign language recognition tasks, often requiring significantly fewer labeled samples [22]. In this approach, the initial layers of the pre-trained model, which have learned to extract general visual features such as edges, textures, and basic shapes, are typically kept frozen or fine-tuned minimally. Only the final layers, responsible for task-specific classification, are retrained or replaced to adapt the model to the nuances of the target sign language dataset. However, ensuring these models generalize well across diverse conditions (lighting, camera angles, user variations, occlusions) remains a key challenge.

Data augmentation techniques, including rotation, flipping, and brightness adjustments, are commonly employed to artificially increase training data diversity and improve model robustness against these variations. Public datasets play a crucial role in training and benchmarking SLR models. One well-known example is the Sign Language MNIST dataset [23], which contains images of static ASL alphabet gestures. Goswami et al. [20] applied a CNN-based approach to classify ASL letters using this dataset, achieving high accuracy. However, datasets like Sign Language MNIST, while useful for initial model development, are often limited in variability (e.g., controlled backgrounds, limited signers) and may not fully capture the complexity and diversity of real-world signing conditions.

A parallel advancement involves pose estimation techniques, notably Google's MediaPipe framework, which enables the real-time, markerless detection of key hand landmarks (keypoints) using deep learning [24,25]. Instead of processing the entire image pixel data, these systems extract geometric features (e.g., landmark coordinates, distances between landmarks, angles formed by landmarks) representing the hand's pose. These features can then be classified using various machine learning models. Multilayer Perceptrons (MLPs), which are fully connected neural networks, are often used for this classification step, mapping the extracted landmark features to specific gesture categories [26,27].

This landmark-based approach offers advantages in computational efficiency and potentially greater robustness to some visual variations (like background changes), making it particularly attractive for mobile deployment [28]. However, landmark detection accuracy can still be sensitive to factors like poor lighting, motion blur, or finger overlap, which can impact the quality of the extracted features. Other traditional classifiers like k-Nearest Neighbors (kNN) [29] and Support Vector Machines (SVMs) [30] have also been applied to landmark data, although neural network models like MLPs often demonstrate better performance, especially with the high-dimensional feature sets derived from detailed landmark data.

Regardless of the approach (direct image classification via CNNs or feature-based classification via landmarks), handling potential class imbalance (where some signs appear more frequently than others in the training data) is crucial for robust performance [31].

Techniques like class weighting during training help mitigate bias towards majority classes, ensuring the model learns to recognize less frequent signs effectively [32]. Evaluating model performance requires metrics beyond simple accuracy. Precision, recall, and F1-score provide more nuanced insights into the model's performance on individual classes, especially with imbalanced data, while confusion matrices help identify specific inter-class confusion patterns [33].

While significant progress has been made, particularly with CNNs for image-based recognition and landmark detection via frameworks like MediaPipe, challenges remain in developing accurate, real-time, and accessible SLR systems. This is especially true for less-resourced languages like LGP and for deployment on computationally constrained mobile platforms. This study builds upon these advancements by implementing and comparing both a direct CNN-based approach and a MediaPipe landmark-based approach within a single mobile application specifically designed for LGP alphabet recognition.

3. Materials and Methods

This section details the materials, methodologies, and implementation steps employed to develop the proposed real-time mobile system for Portuguese Sign Language (LGP) alphabet gesture recognition. The overall approach integrates custom data collection, dual-machine learning model development (CNN-based and landmark-based), mobile application implementation, and systematic evaluation procedures.

3.1. System Architecture

The developed system aims to recognize LGP alphabet letters captured in real time with a mobile device camera and translate them into text. To balance accuracy, computational efficiency, and user flexibility, the system architecture incorporates two distinct recognition pipelines integrated into a single mobile application.

As shown in Figure 1, raw media inputs, including still photographs and video frame extractions, are first funneled through a unified preprocessing module to build the LGP alphabet dataset. During this stage, images are normalized and augmented to produce a balanced, high-variability training corpus. Video sequences are decoded into individual frames, and each frame is cropped around the hand region. Once assembled, the complete dataset is simultaneously fed into two model-training workflows: a CNN branch for direct image-based classification and a MediaPipe+MLP branch that learns from geometric hand landmarks.

In the deployed mobile application, these two trained models operate in parallel (see the right-hand side of Figure 1). The CNN model is embedded locally via TensorFlow Lite, enabling low-latency, offline inference directly on the device's camera stream. In contrast, the MediaPipe+MLP model is hosted on a remote server and accessed over a lightweight REST API: frames are sent to the server for landmark extraction and classification, and the predicted letter is returned to the app. A user-selectable toggle (and an automatic fallback mechanism) allows seamless switching between offline and online modes, adapting in real time to network availability and device performance constraints.

The steps followed in order to implement this system were as follows: (1) acquiring and preprocessing the LGP alphabet dataset, (2) developing and training the machine learning classification models, (3) designing and implementing the mobile application interface and logic, and (4) integrating the models within the mobile application.

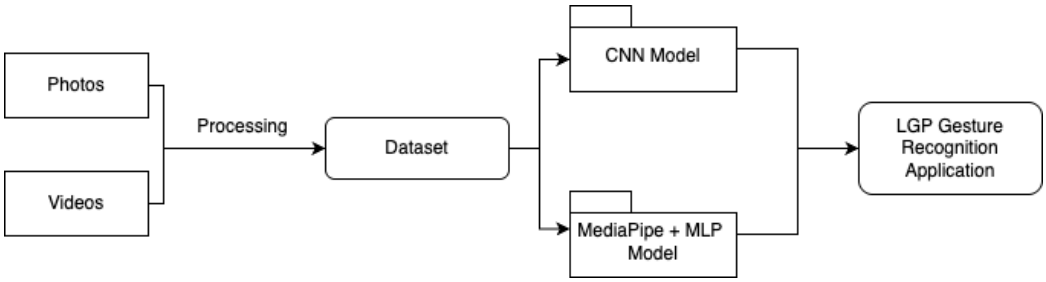

Figure 1. General system architecture: dataset creation from images/videos, training of CNN and MediaPipe+MLP models, and integration into the LGP recognition application.

3.2. LGP Alphabet Dataset: Acquisition and Preprocessing

Developing an effective LGP recognition system requires high-quality, representative training data. As no suitable public dataset for LGP alphabet gestures was available, a custom dataset was created specifically for this study. This involved collecting data from various sources and applying a standardized preprocessing pipeline, as depicted in Figure 2, to ensure data quality and facilitate model training.

This custom dataset comprises two main sources: (1) still images of nine volunteer participants signing all 26 LGP letters in various environments and (2) a "nothing" class sampled from a public Kaggle dataset [34]. Nine participants, each with different physical attributes, environments, and devices, recorded images of the 26 letters of LGP. Each person contributed between 4 and 11 images per letter, using both hands and varying backgrounds. A substantial portion of the images were obtained by extracting frames from videos shot specifically for this project, which increased the size and diversity of the dataset.

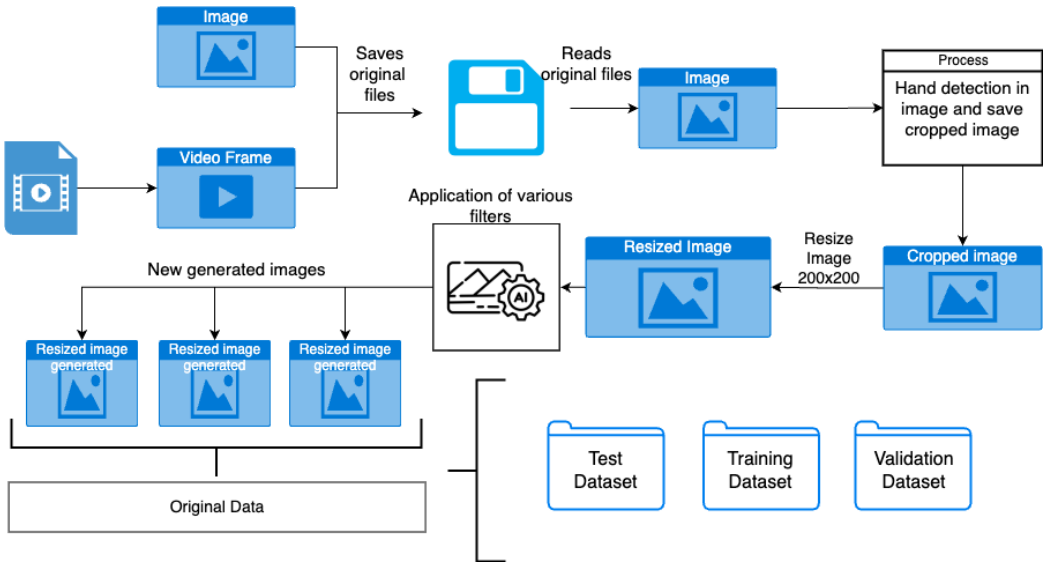

Figure 2. Workflow diagram of the custom LGP alphabet dataset creation process, including data collection, preprocessing, and augmentation.

Our workflow began with saving raw still photos and extracted video frames to disk. Then, each image was loaded into the preprocessing module where MediaPipe located and cropped the hand region. These crops were then resized to a fixed dimension to create our normalized dataset. We applied automatic augmentations to each resized image and combined these variants with the original normalized images to create a diverse sample pool. Finally, we split this pool into training, validation, and test sets.

3.2.1. Data Collection and Preparation

To ensure data diversity and model robustness, images and video sequences of multiple individuals performing LGP alphabet gestures were collected under varying conditions:

- **Direct Image Capture:** Still images were captured from several participants signing LGP letters in diverse environments, featuring different backgrounds, lighting conditions, and camera angles (see examples in Figure 3). This aimed to reflect potential real-world usage scenarios.
- **Video Frame Extraction:** Video recordings of individuals performing sequences of LGP letters provided another data source, significantly increasing the number of available training samples. Relevant frames for each letter gesture were extracted from these videos. To focus the models on the hand gesture itself and minimize background influence, the hand region in each extracted frame was identified using the MediaPipe framework [24], and only this region was retained (see examples in Figure 4). Although many frames extracted this way were visually similar, this apparent redundancy proved beneficial for capturing subtle variations in gesture execution under diverse conditions, potentially improving model robustness. However, the high degree of similarity also presented a risk of model overfitting, reinforcing the necessity for the subsequent application of data augmentation techniques to further diversify the training set.
- **"Nothing" Class Inclusion:** To enable the system to differentiate between a valid LGP letter gesture and the absence of one, a dedicated "nothing" class was included. Images for this class, representing background scenes or hands not performing gestures, were sourced from a publicly available dataset on Kaggle [34] (see an example in Figure 5). This helps reduce false positive classifications when no intended gesture is present.

Figure 3. Examples of collected dataset images for the letter "A" in LGP, captured in different environments to ensure variability.

3.2.2. Data Preprocessing

After data collection, the images underwent preprocessing to ensure they met the quality standards required for training. This process included normalizing image dimensions, adjusting brightness and contrast, and applying data augmentation techniques to improve model generalization. Therefore, the collected images underwent the following standardized preprocessing pipeline before model training:

- **Image Normalization:** All images were resized to a uniform resolution of 200 × 200 pixels. This size was selected as a trade-off between retaining sufficient detail for gesture recognition and maintaining computational efficiency for model training and inference. To preserve the original aspect ratio and avoid distortion during resizing, black padding was added as necessary (see Figure 6).

- **Data Augmentation:** To enhance model robustness and generalization capabilities, data augmentation techniques were applied to the training set, artificially increasing its size and variability. The applied transformations included random rotations (simulating variations in hand orientation), horizontal flips (accounting for potential left-handed signers), brightness and contrast adjustments (simulating different lighting conditions), and small random translations (improving spatial invariance). Examples of augmented images are shown in Figure 7.

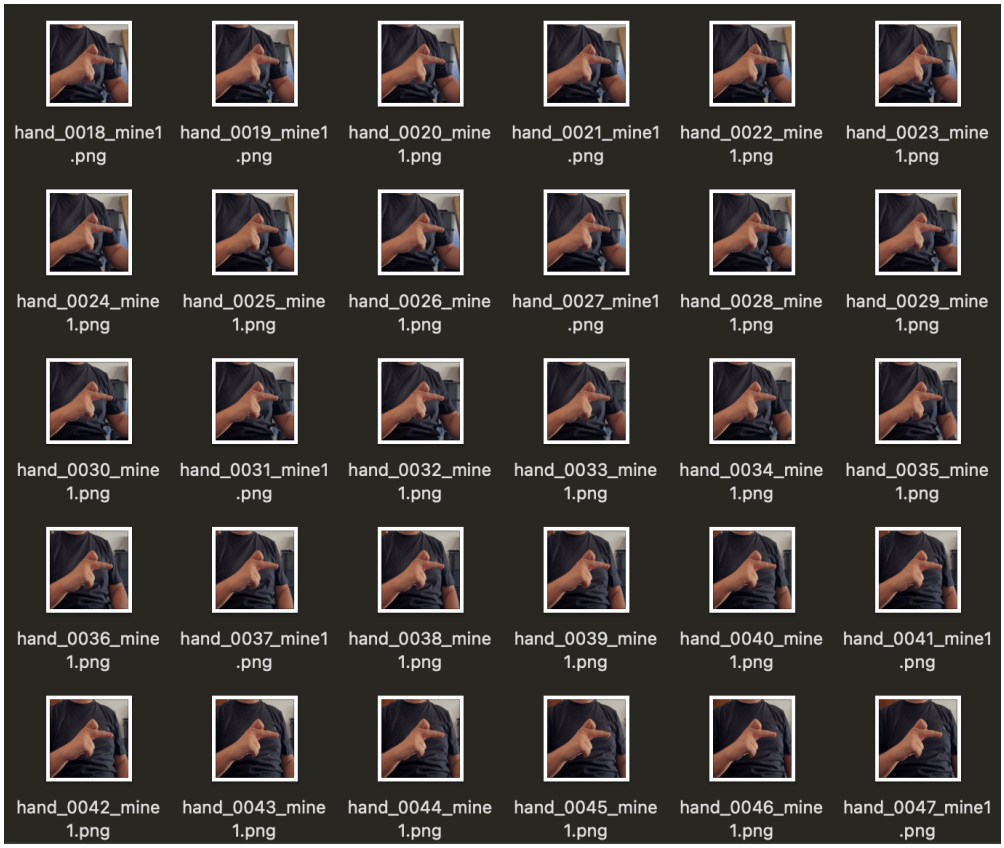

Figure 4. Example frames extracted from a video recording showing the letter "A" in LGP. Hand detection was used to isolate the relevant region.

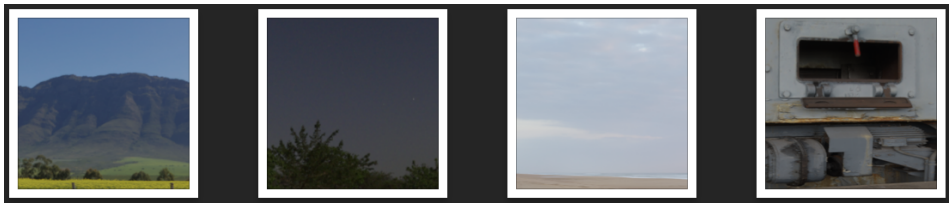

Figure 5. Example image representing the "nothing" class, used to train the models to recognize the absence of a valid gesture.

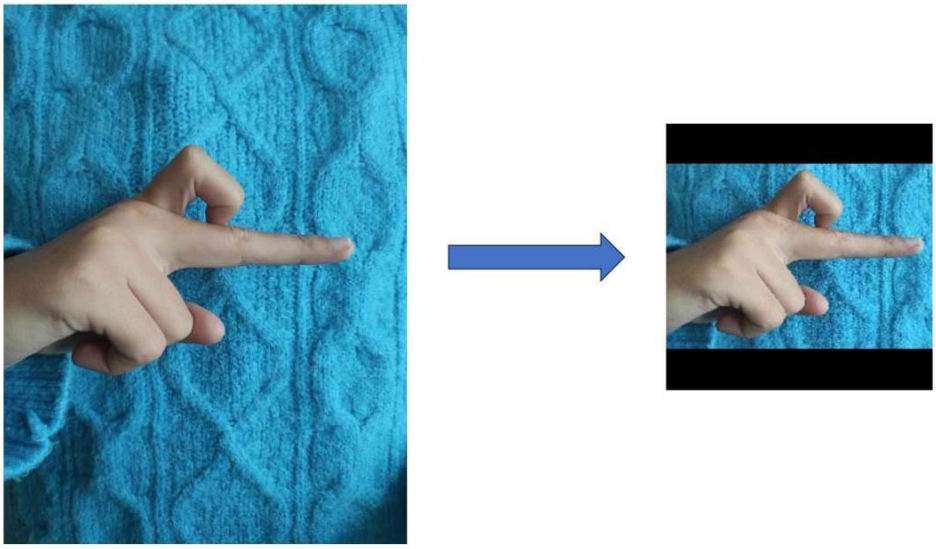

Figure 6. Example illustrating the image normalization process for the letter "A", including resizing to 200 × 200 pixels and padding.

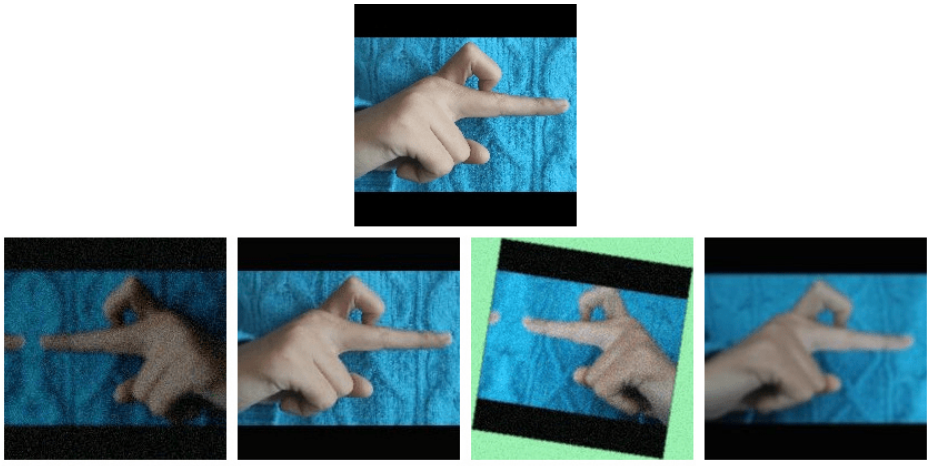

Figure 7. Examples of augmented images generated for the letter "A" using techniques like rotation, flipping, and brightness adjustment.

3.2.3. Dataset Organization and Splitting

The final preprocessed and augmented dataset comprised 27 distinct classes: the 26 letters of the LGP alphabet plus the "nothing" class. To ensure unbiased evaluation and prevent data leakage, the dataset was carefully partitioned into three mutually exclusive subsets: a training set (used for model parameter learning), a validation set (used for hyperparameter tuning and monitoring training progress, e.g., for early stopping), and a test set (used for a final, unbiased evaluation of the trained models' performance). Therefore, after filtering and augmentation, the full dataset was partitioned into

- 94,064 images for training;
- 6157 images for validation;
- 2520 images for testing.

The training set mixes participant photos and video frames. The validation set uses different images of the same signers plus extra variations (ensuring no overlap with training), and the test set entirely comprised completely new recordings under varied conditions to guarantee a realistic, unbiased evaluation.

The distribution of images across these subsets and per class is detailed in Figure 8 and Figure 9, respectively, showing efforts made to maintain a reasonable balance across classes.

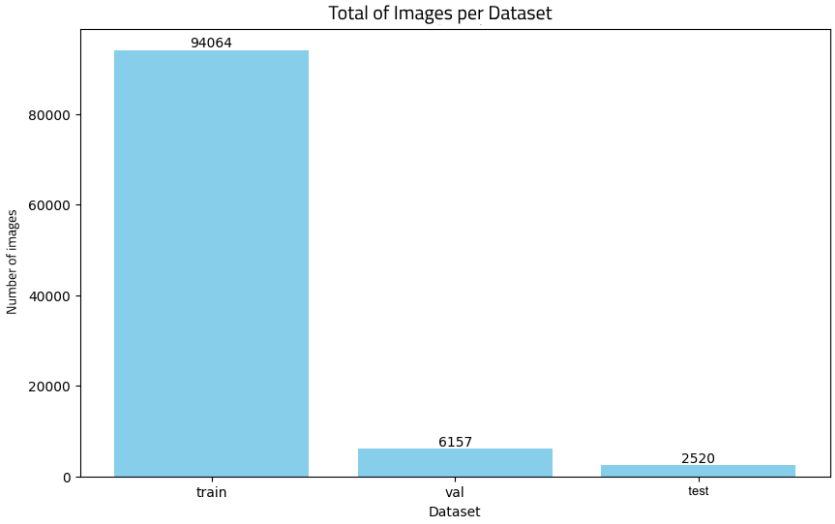

Figure 8. Distribution of images in the custom LGP dataset across the training, validation, and test subsets.

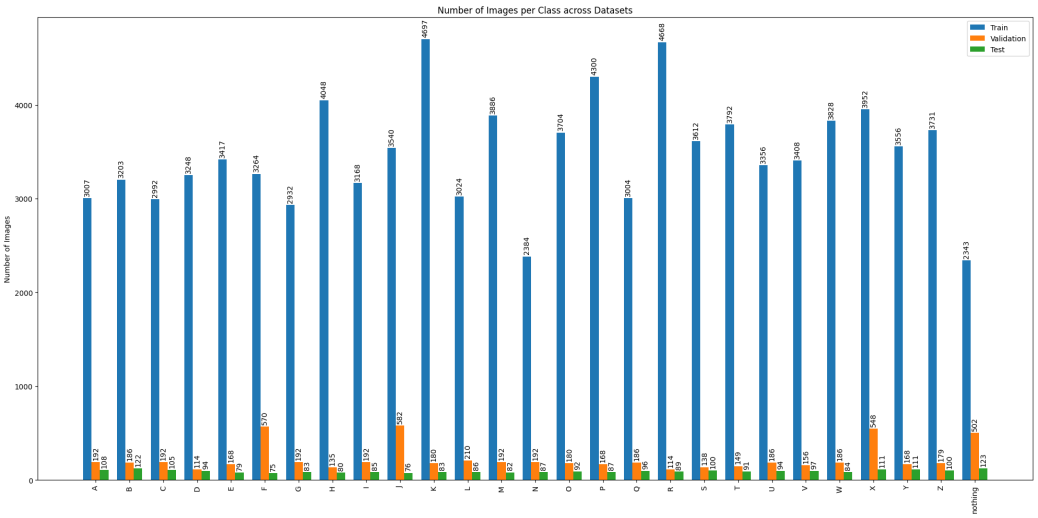

Figure 9. Image count per class within the training, validation, and test subsets of the custom LGP dataset.

3.3. Classification Model Development

Two distinct machine learning models were developed and trained using the prepared LGP dataset, aiming to provide complementary approaches for gesture recognition within the mobile application. The development pipeline for both models is conceptually illustrated in Figure 10.

The first model is based on a Convolutional Neural Network (CNN), leveraging the MobileNetV2 architecture, which was converted to TensorFlow Lite for efficient execution on mobile devices. CNNs are particularly effective in processing visual data, extracting hierarchical features such as edges, shapes, and complex hand configurations. As a result, the model is capable of distinguishing between different letter gestures while maintaining fast inference times. MobileNetV2 was specifically chosen due to its lightweight design, which makes it particularly suitable for execution in mobile environments.

The second model adopts a different approach using the MediaPipe framework to detect 21 hand landmarks and extract geometric features, which are then processed by a

Multilayer Perceptron (MLP). This neural network, composed of fully connected layers, is responsible for mapping the extracted features to specific letter classifications.

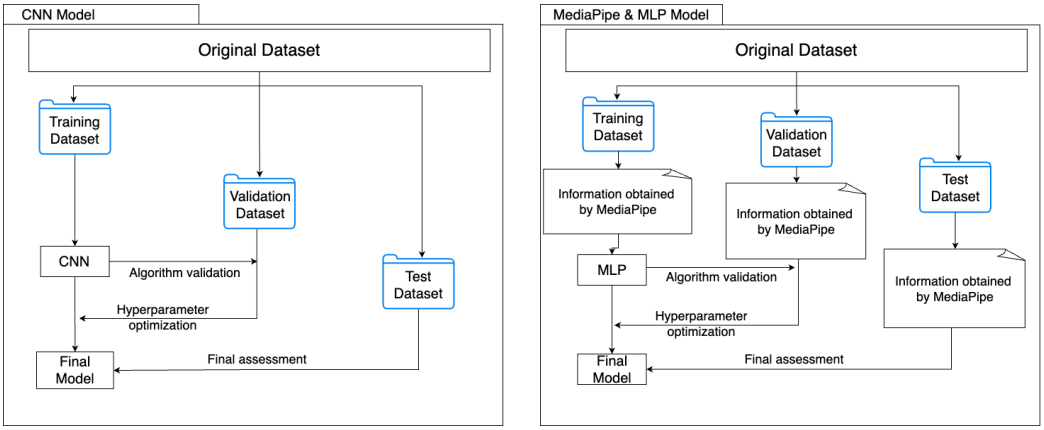

Figure 10. Conceptual diagrams of the two classification approaches: the direct image classification CNN model (on the left); the feature-based MediaPipe landmark extraction followed by MLP classification model (on the right).

The MediaPipe model was implemented as a remote service. However, this approach introduced a dependency on network connectivity and increased latency. Therefore, the system was designed to prioritize local inference through the CNN model whenever possible, ensuring offline functionality and minimizing latency. However, in situations where the CNN model delivered lower classification accuracy, users had the option to switch to the MediaPipe model, which offers more detailed analysis based on hand landmark detection.

The remainder of this subsection details the development of both classification models, covering their architecture and training.

3.3.1. Model 1: CNN for On-Device Image Recognition

The first approach utilizes a Convolutional Neural Network (CNN) for the direct classification of LGP letter gestures from input images. CNNs were chosen due to their proven effectiveness in computer vision tasks, particularly their ability to automatically learn relevant spatial hierarchies of features from pixel data [20].

Architecture

To facilitate efficient execution on resource-constrained mobile devices, the MobileNetV2 architecture [35] was selected as the base model. MobileNetV2 is specifically designed for efficiency, utilizing innovations such as inverted residual blocks to optimize feature extraction without significant computational overhead, making it well suited for mobile platforms. Its favorable balance between classification accuracy and computational cost, combined with its lightweight and modular nature, also makes it particularly suitable for transfer learning.

We employed transfer learning by initializing the convolutional base of MobileNetV2 with weights pre-trained on the large-scale ImageNet dataset [21]. During training for the LGP task, these lower convolutional layers, responsible for extracting general visual patterns (like edges and textures), were kept frozen to retain their learned knowledge. Consequently, only the upper, task-specific layers were fine-tuned for the specific goal of recognizing the 27 LGP classes (26 letters + "nothing").

Following the frozen MobileNetV2 base, a classification head was added. This head begins with a Global Average Pooling (GAP) layer, which significantly reduces the number of parameters compared to a Flatten layer while retaining spatial information, thus helping to prevent overfitting and reduce computational load. This was followed by two fully

connected (dense) layers with 128 and 64 neurons, respectively, responsible for learning higher-level combinations of features extracted by the base model and mapping them towards the final classification.

ReLU activation functions were used after these dense layers to introduce non-linearity. To promote stable gradient flow during training, He-Normal initialization was used for the weights of these layers. Furthermore, Batch Normalization was applied after each dense layer to accelerate convergence and further mitigate the risk of overfitting. The final layer employed a Softmax activation function to output probabilities for each of the 27 target classes. The overall architecture, integrating the pre-trained base and the custom classification head, is depicted in Figure 11.

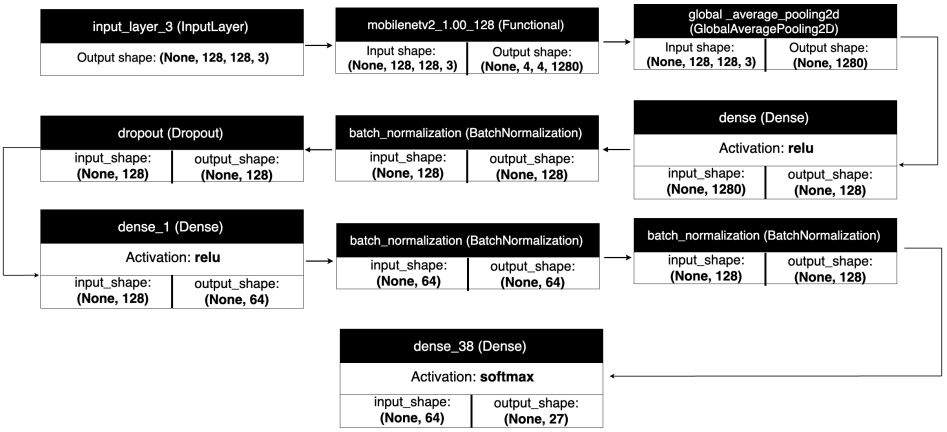

Figure 11. Architecture diagram of the CNN model, utilizing a pre-trained MobileNetV2 base followed by custom classification layers (Global Average Pooling, dense layers).

Training

The CNN model was trained using the custom LGP dataset (Section 3.2) for a maximum of 100 epochs. The Adam optimizer was employed with an initial learning rate of 0.001 and a batch size of 32. To prevent overfitting and optimize training time, early stopping was implemented, halting training if the validation loss did not improve for a predefined number of consecutive epochs. Additionally, a learning rate scheduler was used to dynamically decrease the learning rate when the validation loss plateaued, allowing for finer adjustments during later stages of training.

Class Imbalance Handling

Initial training revealed lower accuracy for certain visually similar or less frequent classes (e.g., “G”, “X”, “F”, “L”). To address this potential class imbalance, class weights were incorporated into the loss function during training. Weights were assigned inversely proportional to the observed class accuracy from initial runs, effectively increasing the penalty for misclassifying underperforming classes. Equation (1) shows the initial calculation, although manual adjustments were sometimes made to further boost weights for persistent problematic classes. This strategy significantly improved the balanced accuracy across all classes.

$$\text{Weight}_c \approx \frac{1}{\text{Accuracy}_c} \tag{1}$$

where Weight_c and Accuracy_c are the weight and observed accuracy for class c, respectively.

While the CNN model provided a solid initial performance baseline, difficulties arose in distinguishing visually ambiguous gestures, such as “G” and “X”. This suggests potential limitations in relying solely on pixel-based features for fine-grained discrimination. Therefore, to specifically address these classification challenges, a second model was developed using MediaPipe to extract and analyze geometric hand landmark features.

Mobile Deployment

After training, the model was converted to the TensorFlow Lite (TFLite) format. This conversion optimizes the model for mobile deployment by reducing its size and enabling efficient inference using the TFLite interpreter on Android and iOS devices, facilitating the offline recognition mode of the application.

3.3.2. Model 2: MediaPipe Landmarks and MLP Classification

The second approach leverages hand landmark detection as an intermediate feature extraction step, followed by classification using a Multilayer Perceptron (MLP). This method aims to potentially improve robustness by focusing explicitly on hand geometry rather than raw pixel data, while also exploring a different computational paradigm (feature extraction + simpler classifier).

Architecture and Feature Extraction

Google's MediaPipe gesture recognizer [24,25,28] was used to detect 21 key 3D landmarks (keypoints) on the hand in real time from the camera feed (Figure 12). These landmarks represent joints and fingertips. Instead of using the raw image, features were derived from these landmark coordinates. These included the normalized (x, y, z) coordinates of each landmark relative to the wrist and potentially additional engineered features, such as distances between specific landmarks (e.g., fingertip to wrist, fingertip to fingertip) and angles between key joint vectors, aiming to capture finer geometric details of the hand pose. These normalized and engineered features formed the input vector for the subsequent classifier.

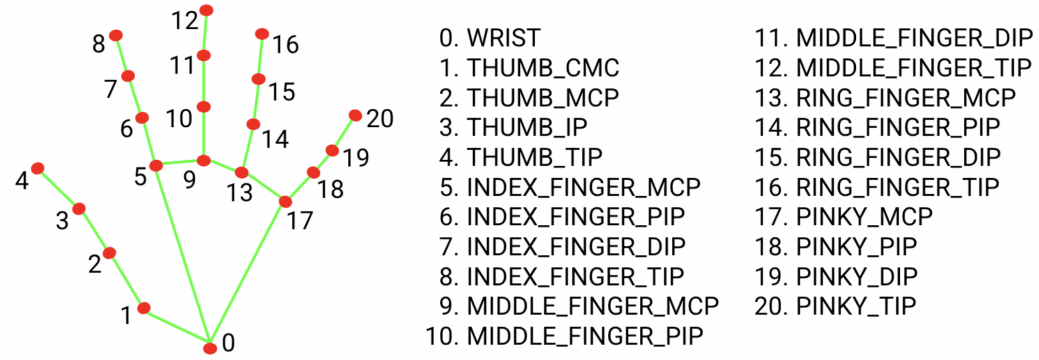

Figure 12. The 21 hand landmark points and their corresponding indices as detected by MediaPipe's hand landmark model [25].

MLP Classifier

An MLP was trained to classify the LGP alphabet gestures based on the extracted landmark features. The MLP architecture consisted of multiple fully connected layers with ReLU activation functions, designed to learn complex patterns from the geometric hand pose data. Following experimentation and refinement, the final architecture employed layers with 256, 128, and 64 neurons, followed by dropout layers after each hidden layer to mitigate overfitting. The output layer used Softmax activation for the 27 classes. The conceptual architecture is shown in Figure 13. Training procedures, including the use of the Adam optimizer, learning rate scheduling, early stopping, and class weighting (similar to Model 1), were also applied.

These architectural choices and training adjustments proved effective, significantly improving the model's generalization ability, particularly for distinguishing challenging gestures previously misclassified by simpler configurations. Overall, this MediaPipe feature extraction followed by MLP classification provided an alternative solution that, as detailed

later in the results, offered advantages in classification accuracy for certain complex gestures compared to the direct CNN model.

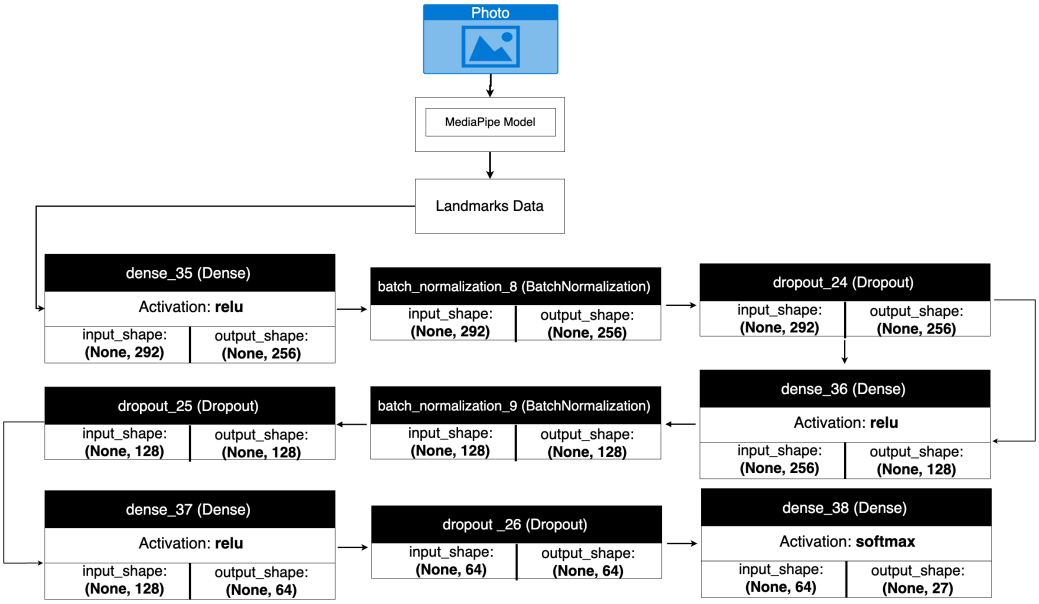

Figure 13. Conceptual diagram of the MLP classifier architecture used with MediaPipe landmark features.

3.4. Mobile Application Implementation and Interface

To provide a user-friendly interface for the LGP letter recognition system, a mobile application was developed using Flutter. This framework was selected for its ability to build natively compiled applications for both Android and iOS from a single codebase, alongside its performance characteristics and flexible UI design features. The application's core functionality centers on real-time gesture recognition: it utilizes the device's camera stream as input, processes the video frames using one of the selected recognition models, and displays the corresponding recognized LGP letter as text output to the user with minimal latency.

A key aspect of the application design is the integration and user-selectable switching between the two distinct recognition models developed in this study. The locally executed TFLite CNN model enables offline gesture recognition, while an interface to the remotely hosted MediaPipe+MLP model provides an alternative online recognition pathway. A toggle switch within the UI allows users to easily select their preferred mode based on factors like network availability, desired speed, or potential accuracy differences between the models.

The implementation of the MediaPipe+MLP model required a remote service architecture. This decision stemmed from the fact that the official MediaPipe plugin for Flutter [36], while offering integration possibilities, does not currently support the computer vision tasks available in the MediaPipe framework. Specifically, tasks crucial for this study, such as the Gesture Recognizer pipeline, were not directly executable on-device via the Flutter plugin at the time of development. This limitation necessitated processing this recognition on a remote server accessed by the mobile application.

To overcome this, a server-side API was implemented using Python 3.13.2 and the Flask framework. This API receives image frames from the mobile app, performs the MediaPipe landmark detection and feature processing, classifies the gesture using the trained MLP model, and transmits the resulting recognized letter back to the mobile application. While enabling the use of the landmark-based approach, this remote processing inherently introduces network dependency and potential latency variations.

Beyond the real-time translation feature, the application includes supplementary functionalities to enhance user experience and utility. An interactive learning mode allows users to browse through the LGP alphabet letters and view illustrations or examples of their corresponding standard gestures. Additionally, a text-to-gesture translation feature was incorporated, enabling users to input text and see the corresponding sequence of LGP alphabet signs. The inherent offline capability provided by the local TFLite model ensures the application remains functional even without an internet connection when that mode is selected. The overall UI, examples of which are shown in Figure 14, was designed with an emphasis on intuitiveness and responsiveness to facilitate ease of use.

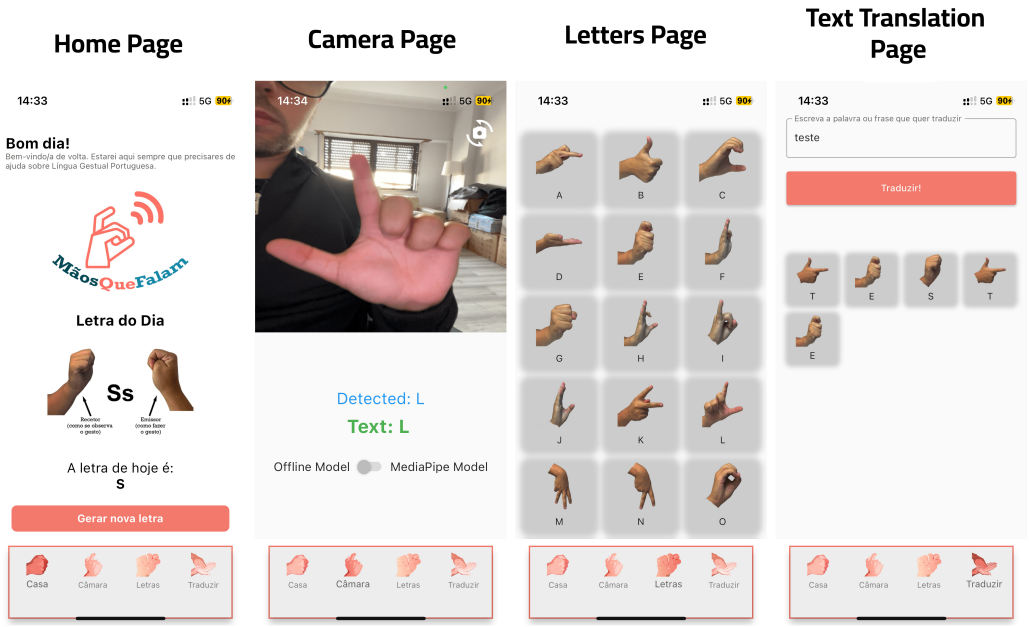

Figure 14. Screenshots illustrating the main user interface pages of the developed mobile application, including the live translation view and alphabet learning components.

3.5. Evaluation Methodology

The performance and usability of the developed system were assessed through distinct evaluation procedures.

3.5.1. Model Performance Evaluation

The classification performance of both the CNN and the MediaPipe+MLP models was quantitatively evaluated using the dedicated test subset of the custom LGP dataset to ensure an unbiased assessment of their classification ability. Standard machine learning metrics were employed for this evaluation. Overall accuracy provided the percentage of correctly classified gestures across all classes. However, to gain more nuanced insights into model behavior, particularly considering potential class imbalances [33], performance was further assessed using precision (the proportion of gestures classified as a specific letter that were truly that letter, important for minimizing false positives), recall or sensitivity (the proportion of actual instances of a letter that were correctly identified, important for minimizing missed gestures), and the F1-score (the harmonic mean of precision and recall, offering a single balanced metric).

Furthermore, confusion matrices were generated to visually analyze the classification results, helping to identify specific problematic gesture pairs and pinpoint areas for model improvement. This comprehensive set of metrics allowed for a thorough evaluation of the models' effectiveness, providing insights crucial for understanding performance limitations and informing necessary optimization strategies. In addition to classification

performance, inference speed (latency) was also measured and considered for both models under different operational conditions, such as local versus remote execution and variations in device hardware. The specific results obtained from applying these evaluation metrics are presented and discussed in Section 4.1.

3.5.2. Usability Evaluation

Beyond quantitative model performance, the overall usability and user experience (UX) of the mobile application were also assessed qualitatively to gain insights into its practical application by end-users. Structured interviews and observation sessions were conducted with real users.

Considering accessibility as a fundamental requirement for this type of application, the primary goal of this evaluation was to ensure the interface was intuitive and user-friendly. The qualitative evaluation focused specifically on assessing the application's interface clarity and ease of navigation, the intuitiveness and practicality of the core real-time translation feature, and the system's responsiveness under different conditions, including users' perception of latency.

Furthermore, the perceived usefulness of auxiliary features, such as the integrated learning mode and the ability to switch between recognition models, was investigated. The assessment aimed to capture overall user satisfaction while identifying both strengths and areas for potential enhancement, ensuring the developed application is not only technically functional but also practical for its intended purpose as an assistive communication tool. The findings from this usability evaluation are further explored in Section 4.2.

4. Results

The proposed system was evaluated from two key perspectives: the performance of the gesture recognition models and the user experience with the mobile application. The evaluation begins with an assessment of the machine learning models used to classify LGP letters, including a comparative analysis of their performance, identification of challenging classes, and a benchmark against previous work. This is followed by a discussion of the models' integration into the mobile application, highlighting the challenges encountered and the results achieved in terms of prediction accuracy, processing speed, and overall user satisfaction.

4.1. Evaluation of Classification Models

The classification performance of both the CNN and the MediaPipe+MLP models was quantitatively evaluated using the dedicated test subset of the custom LGP dataset (details in Section 3.2.3) to ensure an unbiased assessment. Standard machine learning metrics were employed, including accuracy, precision, recall, F1-score, and confusion matrices.

4.1.1. Model 1: CNN for Letter Recognition

The CNN model, based on the MobileNetV2 architecture and optimized with TensorFlow Lite for mobile deployment, was trained for 32 epochs before early stopping criteria were met based on validation loss. Analysis of the training and validation curves (Figure 15) revealed a rapid increase in training accuracy, approaching 100%, while the validation accuracy plateaued relatively early at approximately 61.57%. This significant gap indicates substantial overfitting, where the model learned the training data specifics effectively but struggled to generalize to unseen validation data. Although regularization techniques like dropout and learning rate decay were applied during training (Section 3.3.1), they did not fully mitigate this issue.

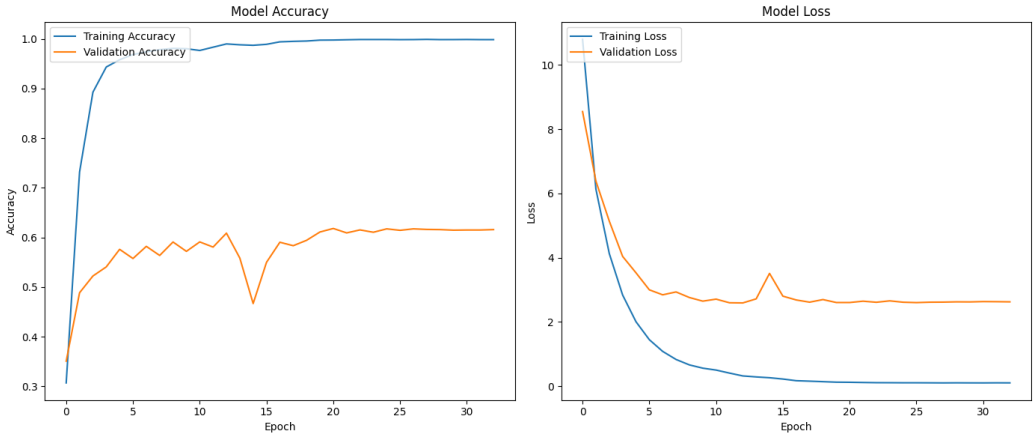

Figure 15. Accuracy and loss curves for the CNN model during training, showing training (blue) and validation (orange) metrics over 32 epochs. The divergence highlights the overfitting observed.

The confusion matrix for the CNN model on the test set (Figure 16) provides a detailed view of class-specific performance. While simpler gestures like "A" and "B" were classified with high accuracy, significant confusion occurred between visually similar gestures. Notably, the model frequently misclassified "X" (often confused with "V" or "P") and "G". This suggests difficulty in capturing the fine-grained finger positioning details required for these ambiguous gestures using the purely image-based approach.

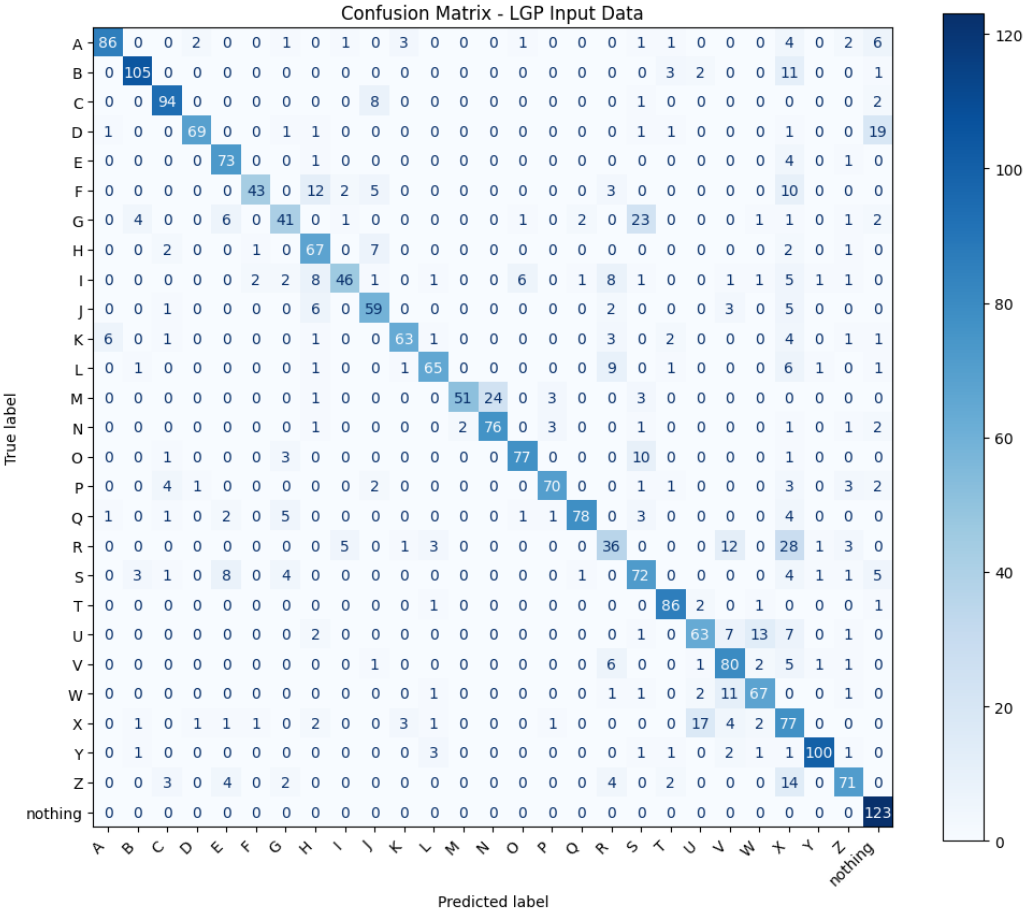

Figure 16. Confusion matrix for the CNN model on the test set, visualizing correct classifications and inter-class errors.

Quantitatively, the CNN model achieved an overall accuracy of 76% on the test set, as detailed in the classification report. The macro-averaged precision was 79%, recall was 75%, and F1-score was 76%. These overall metrics, however, mask significant performance variations across classes. For instance, class "G" exhibited a very low recall of 36%, indicating the model failed to identify a large proportion of its instances. Similarly, class "R" had a low precision of 33%. Letters such as "X" (F1 = 0.59) and "F" (F1 = 0.67) also showed weaker performance compared to well-recognized letters like "Y" (F1 = 0.91) or "Q" (F1 = 0.89).

Although the model achieved reasonable accuracy, it encountered difficulties in correctly identifying certain gestures, particularly those with subtle variations in finger positioning. These challenges indicate that expanding the training dataset or adjusting the model architecture could enhance performance, especially for the most problematic classes.

4.1.2. Model 2: MediaPipe for Hand Landmark Detection

The second model, combining MediaPipe landmark extraction with an MLP classifier, was trained for 22 epochs. The training and validation curves (Figure 17) showed improved stability compared to the CNN, with the final validation accuracy reaching 93.17%. The smaller gap between training and validation accuracy suggests better generalization for this landmark-based approach, although some degree of overfitting remained.

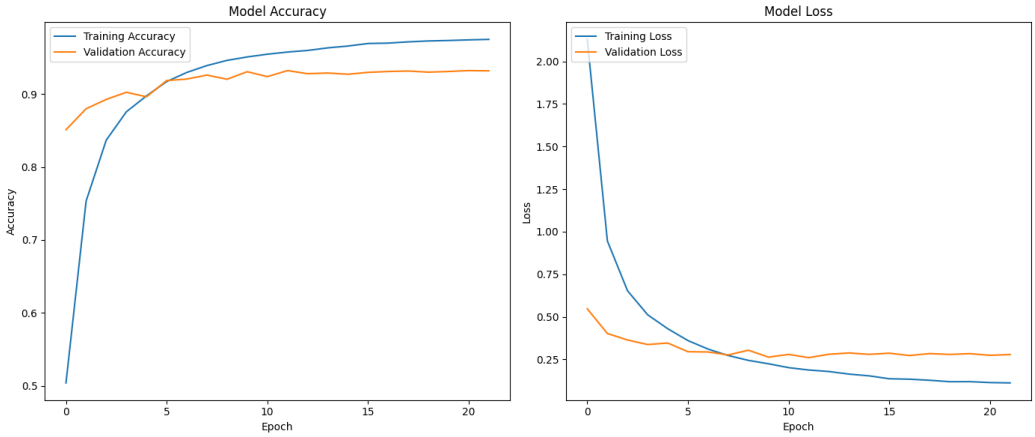

Figure 17. Accuracy and loss curves for the MediaPipe+MLP model during training, showing training (blue) and validation (orange) metrics over 22 epochs.

Analysis of the confusion matrix (Figure 18) indicates strong performance for many letters (e.g., "A", "B", and "O"). However, difficulties persisted for visually confusable gestures, particularly "X" and "K", which were often confused with each other or similar shapes like "V" and "P". The extraction of geometric features aided overall robustness, but differentiating minute finger configuration differences for these specific letters remained challenging. It is also important to note that predictions of the "nothing" class sometimes occurred for actual letter gestures; this typically indicated a failure of the underlying MediaPipe hand detection module (due to poor lighting, occlusion, etc.) rather than a classification error based on valid landmarks. Thus, the lower sample count in (Figure 18) results from including only those images for which both detection and classification were successfully completed. This approach allows for a more accurate assessment of the classifier's capabilities, independent of errors outside its scope.

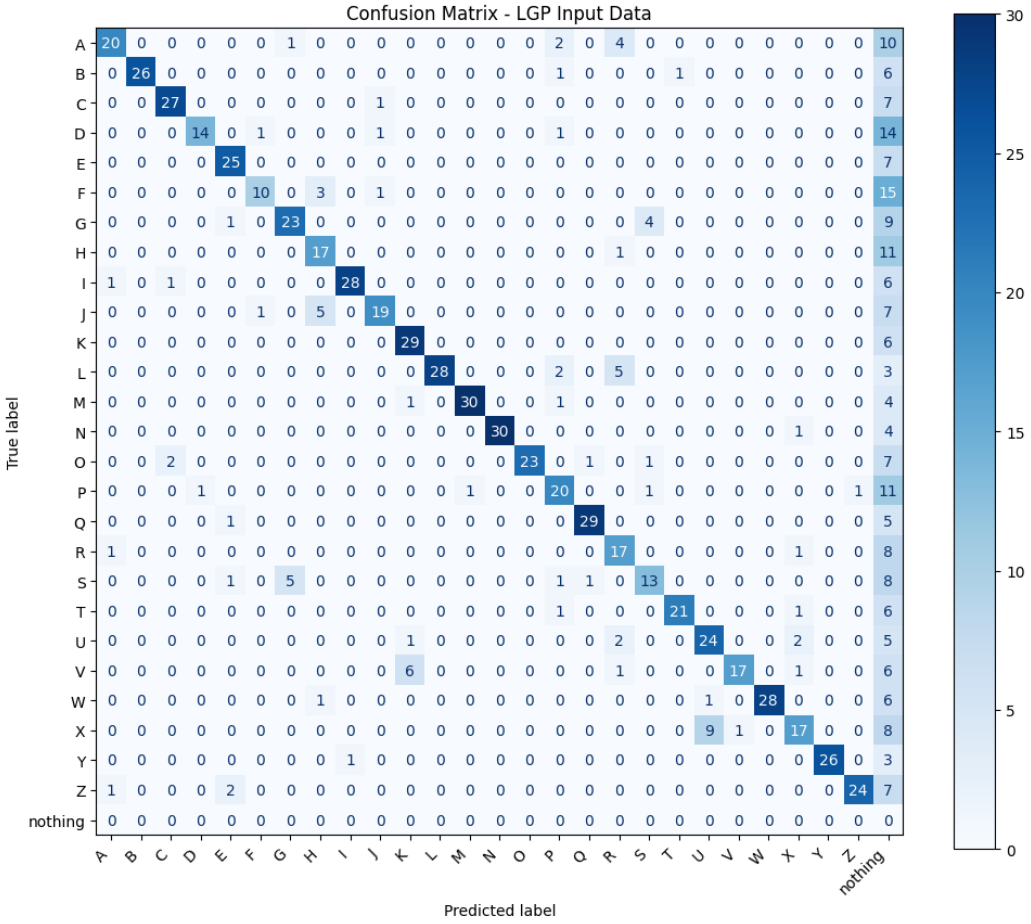

Figure 18. Confusion matrix for the MediaPipe+MLP model on the test set.

The MediaPipe model showed a mean precision of 77%, with a global F1-score of 77%. These numbers demonstrate that the model achieved a good balance between precision and recall, proving more robust than the CNN model across several classes. For example, class "B" had a precision of 96% and an F1-score of 88%, reflecting the model's ability to correctly identify this letter in most instances. On the other hand, class "X" continued to be one of the most problematic, with an F1-score of only 0.52, revealing that the model had difficulty distinguishing this gesture from other similar ones, such as "V" and "K". Class "K", in turn, showed inferior performance when compared to the CNN model, with an F1-score of 56%, confirming that the model failed to correctly identify a significant proportion of instances of this letter.

One notable limitation of the model was its handling of the "nothing" class, which was often predicted when the system failed to detect hand landmarks in the image. This issue typically occurred under poor lighting conditions or when parts of the hand were occluded, negatively impacting overall performance. Even so, the model showed greater robustness in recognizing complex gestures than the CNN-based alternative, thanks to the explicit extraction of geometric features from hand poses.

4.1.3. Comparative Analysis

A direct comparison of the overall performance metrics (Table 1) confirms that the MediaPipe+MLP model achieved a slightly higher overall accuracy and F1-score (77%) than the CNN model (76%). This reflects the landmark-based model's better performance on several challenging classes and superior generalization observed during training.

Table 1. Comparison of overall classification results between the CNN and MediaPipe+MLP models on the test set.

Model	Accuracy	Macro Avg. Precision	Macro Avg. Recall	Macro Avg. F1-Score
CNN	76%	79%	75%	76%
MediaPipe+MLP	77%	78%	76%	77%

However, the primary distinction lies in their operational characteristics. The local CNN (TFLite) provided significantly lower latency (1–2 s on high-end devices), crucial for real-time usability, and offered essential offline functionality. In contrast, the remote MediaPipe+MLP model, while demonstrating higher accuracy and robustness for complex gestures, suffered from higher latency (3–5 s, network dependent) and required internet connectivity. This presents a clear trade-off: the CNN prioritizes speed and accessibility, while the MediaPipe+MLP prioritizes raw classification accuracy.

Compared to earlier LGP recognition works relying on specialized hardware like Kinect [17,18], which faced resolution limitations for fine finger details, this study demonstrates the viability of using standard mobile cameras with modern deep learning. Our approach integrates and compares two distinct strategies: direct image recognition using a CNN, and detailed 3D landmark analysis through MediaPipe enhanced with the calculation of angles and distances between landmarks to capture subtle variations in finger positioning. Unlike Takashi's approach that relies solely on 3D landmarks [27], our solution offers greater flexibility depending on the application context and provides valuable insights into adapting different techniques for mobile-based LGP recognition.

4.2. Mobile Application Evaluation

The usability and practical performance of the mobile application, which integrates both recognition models, were evaluated through testing on various devices and structured user interviews. Developed in Flutter to ensure compatibility across both Android and iOS platforms, the application offers a practical solution for real-time LGP gesture recognition. Its assessment focused on both technical performance and user experience, with tests conducted under a range of conditions, including varying lighting environments, camera angles, and device capabilities, in order to evaluate the system's adaptability and robustness.

4.2.1. Technical Performance Within Application

Real-world testing confirmed the distinct operational characteristics and latency profiles of the two models within the mobile application. The local TFLite CNN model provided rapid responses, typically within 1–2 s on high-end devices (e.g., iPhone 13 mini, iPhone 15, and iPhone 16), facilitating real-time interaction and offering the significant advantage of offline functionality. However, on lower-end devices with less processing power (e.g., Samsung Galaxy A52, Xiaomi Redmi Note 13, and an older Android device), its inference speed slowed noticeably into the 3–5 s range. This slower execution speed on less capable hardware hindered the perceived responsiveness and real-time feel, impacting its overall reliability for fluid communication in those scenarios, even if recognizing simpler gestures like "B", "C", and "L" remained relatively successful compared to complex ones like "G", "X", and "P". The challenges in recognizing these complex gestures might be attributed both to the model's inherent difficulties and potentially to precision reductions introduced during the TFLite conversion process, which optimizes the model for mobile deployment but can impact fine-grained discrimination.

Conversely, the remote MediaPipe+MLP model generally exhibited response times between 3 and 5 s, largely independent of the device's processing power but heavily influenced by network quality and stability. While requiring an internet connection and introducing potential latency, this pathway consistently delivered higher accuracy in recognizing complex and visually similar gestures across different devices, provided the connection was stable. Communication with the Flask server ensured consistent processing logic, although network variability could lead to fluctuating response times, posing challenges in unstable network environments where processing could be significantly delayed. The reliability of this model was also influenced by the device's camera quality, as lower-quality cameras sometimes impacted the initial MediaPipe landmark detection, consequently affecting the final classification accuracy.

Overall, the testing highlighted that the CNN model functioned as the more responsive but potentially less accurate offline solution, particularly suitable for capable devices, while the MediaPipe model offered greater accuracy and robustness, especially for difficult gestures, but was dependent on network connectivity and exhibited higher latency. The model-switching feature within the application functioned as intended, allowing users to manually select between the faster, offline CNN and the more accurate, online MediaPipe+MLP model based on their immediate needs, prioritizing either speed or accuracy. Furthermore, an automatic fallback mechanism, reverting to the local TFLite model if the remote server connection failed, successfully ensured continuous operation.

4.2.2. User Experience Feedback

Qualitative user experience feedback was gathered through structured interviews with 10 participants who tested the application on mobile devices with varying capabilities. The primary goal was to assess the application's practical usability and identify areas for enhancement. Participants generally reported the application interface to be intuitive and easy to navigate. The functionality allowing users to switch between the offline (faster) and online (more accurate) recognition models was unanimously considered a valuable feature, providing useful adaptability to different conditions (e.g., network availability).

Despite the positive overall reception, specific areas for improvement were highlighted. A significant majority (70%, 7 out of 10 users) pointed out the lack of clear feedback when the system failed to recognize a gesture, which could lead to user confusion. They suggested incorporating messages to guide users, for example, advising adjustments to hand positioning or environmental lighting. Additionally, 20% (2 users) experienced difficulty identifying some icons within the bottom navigation bar, recommending the addition of text labels or the use of more universally understood symbols. Minor suggestions also included optimizing the visual presentation of example gestures within the learning module for better clarity (e.g., removing shadows), particularly on lower-resolution screens.

5. Conclusions

This study successfully developed and evaluated a mobile application for real-time Portuguese Sign Language (LGP) alphabet-to-text translation, addressing a critical need for accessible communication tools. By implementing and comparing two distinct machine learning approaches in the form of an on-device Convolutional Neural Network (CNN) and a remote MediaPipe-based Multilayer Perceptron (MLP), this research demonstrated a practical hybrid system that balances recognition speed, accuracy, and operational flexibility for LGP users.

The evaluation confirmed the complementary nature of the deployed models: the CNN (MobileNetV2 with TensorFlow Lite) offered rapid, offline inference (1–2 s on high-end devices, 76% F1-score), enhancing accessibility, while the remote MediaPipe+MLP

approach provided slightly higher overall accuracy and robustness for complex gestures (3–5 s, 77% F1-score), though dependent on network connectivity. Key contributions of this work include the development of this dual-model mobile architecture featuring an automatic switching mechanism, the creation of a novel LGP alphabet dataset vital for training and validation, and a direct comparative analysis of image-based versus landmark-based recognition strategies within a mobile LGP context. Socially, the application demonstrates potential as a valuable tool for improving communication and aiding LGP alphabet learning.

Despite these achievements, limitations were identified. Neither model achieved optimal accuracy across all LGP alphabet gestures, with specific visually similar letters proving challenging for both (e.g., 'G', 'X', and 'R' were the most challenging for the CNN model and 'X' and 'K' were the most challenging for the MediaPipe+MLP approach). The CNN model exhibited overfitting and sensitivity to device capabilities, while the MediaPipe model's performance was influenced by network stability and the precision of landmark detection. The custom dataset, while foundational, was limited to alphabet gestures and may not fully capture all real-world signing variations. User feedback also indicated a need for more intuitive error handling within the application.

Future work should focus on refining model accuracy, particularly for problematic gestures, potentially through advanced deep learning architectures, more sophisticated feature engineering, and further expansion of the training dataset to include greater diversity. A significant advancement would be the system's expansion to recognize continuous LGP signing, moving beyond isolated letters to interpret words and simple phrases. Further technical optimizations, exploration of adapting the methodology for other sign languages, and extensive usability studies with the deaf community are crucial next steps. This research provides a solid foundation for ongoing development in assistive communication technology, aiming to enhance accessibility and foster greater inclusion for the deaf community in Portugal and possibly for other sign languages.

Author Contributions: Conceptualization, G.F., G.M., and R.J.; methodology, G.F., G.M., and R.J.; software, G.F.; validation, G.F., G.M., P.A.S., and R.J.; formal analysis, G.M., and R.J.; investigation, G.F., G.M., P.A.S., and R.J.; resources, G.F., G.M., and R.J.; writing—original draft preparation, G.F.; writing—review and editing, G.F., G.M., P.A.S., and R.J.; supervision, G.M. and R.J.; project administration, R.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Ethical review and approval were waived for this study due to the fact that participants were only asked to experiment with a mobile application and give their opinion anonymously. All participants agreed to take part on a voluntary basis.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study, and participation in the study was voluntary. By agreeing to participate in the study, the participants gave their consent. The data were collected through interviews, and only a few statistics on the anonymous data are presented in the article.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author, subject to copyright restrictions and permission from the relevant sources.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lucas, C. (Ed.) *Sociolinguistic Variation in American Sign Language*; Cambridge University Press: Cambridge, UK, 2001. [CrossRef]
2. Klima, E.S.; Bellugi, U. *Signs of Language*; Harvard University Press: London, UK, 1979.
3. Stokoe, W. Sign Language Structure: An Outline of the Visual Communication Systems of the American Deaf. *J. Deaf. Stud. Deaf. Educ.* **2005**, *10*, 3–37. [CrossRef]

4. Assembleia da República. Lei Constitucional n.º 1/97, de 20 de setembro. Point (h) of Paragraph 2 of Article 74. 1997. Available online: https://diariodarepublica.pt/dr/detalhe/lei-constitucional/1-653562 (accessed on 22 July 2024)
5. Pfau, R.; Steinbach, M.; Woll, B. (Eds.) *Sign Language: An International Handbook*; De Gruyter: Berlin, Germany, 2012. [CrossRef]
6. Tamura, S.; Kawasaki, S. Recognition of sign language motion images. *Pattern Recognit.* **1988**, *21*, 343–353. [CrossRef]
7. Tanibata, N.; Shimada, N.; Shirai, Y. Extraction of Hand Features for Recognition of Sign Language. In Proceedings of the International Conference on Vision Interface, New York, NY, USA, 8–10 July 2002.
8. Guerin, G. Sign Language Recognition—Using MediaPipe & DTW. Available online: https://data-ai.theodo.com/en/technical-blog/sign-language-recognition-using-mediapipe (accessed on 22 July 2024).
9. Li, G.; Tang, H.; Sun, Y.; Kong, J.; Jiang, G.; Jiang, D.; Tao, B.; Xu, S.; Liu, H. Hand gesture recognition based on convolution neural network. *Clust. Comput.* **2019**, *22*, 2719–2729. [CrossRef]
10. Papastratis, I.; Dimitropoulos, K.; Konstantinidis, D.; Daras, P. Continuous Sign Language Recognition Through Cross-Modal Alignment of Video and Text Embeddings in a Joint-Latent Space. *IEEE Access* **2020**, *8*, 91170–91180. [CrossRef]
11. Cihan Camgoz, N.; Koller, O.; Hadfield, S.; Bowden, R. Sign Language Transformers: Joint End-to-End Sign Language Recognition and Translation. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, 14–19 June 2020; pp. 10020–10030. [CrossRef]
12. Tayade, A.; Halder, A. Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning. *Int. J. Res. Publ. Rev.* **2021**, *2*, 9–17. [CrossRef]
13. Indriani; Harris, M.; Agoes, A.S. Applying Hand Gesture Recognition for User Guide Application Using MediaPipe. In Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021), Online, 12 October 2021; Atlantis Press: Amsterdam, The Netherlands, 2021; pp. 101–108. [CrossRef]
14. Feng, Q.; Yang, C.; Wu, X.; Li, Z. A smart TV interaction system based on hand gesture recognition by using RGB-D Sensor. In Proceedings of the 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), Shengyang, China, 20–22 December 2013; pp. 1319–1322. [CrossRef]
15. Mora-Zarate, J.E.; Garzón-Castro, C.L.; Castellanos Rivillas, J.A. Construction and Evaluation of a Dynamic Sign Dataset for the Colombian Sign Language. In Proceedings of the 2024 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Bogota, Colombia, 13–15 November 2024; pp. 1–5. [CrossRef]
16. Neiva, I.G.S. Desenvolvimento de um Tradutor de Língua Gestual Portuguesa. Master's Thesis, NOVA School of Science and Technology, NOVA University Lisbon, Lisbon, Portugal, 2014. Available online: http://hdl.handle.net/10362/14753 (accessed on 22 July 2024).
17. Ribeiro, P.R. Sistema de Reconhecimento de Língua Gestual Portuguesa Recorrendo à Kinect. Master's Thesis, School of Engineering, University of Minho, Minho, Portugal, 2019. Available online: https://hdl.handle.net/1822/72165 (accessed on 22 July 2024).
18. Oliveira, O.R.S.A. Tradutor da Língua Gestual Portuguesa Modelo de Tradução Bidireccional. Master's Thesis, ISEP—Porto School of Engineering, Polytechnic University of Porto, Porto, Portugal, 2013. Available online: http://hdl.handle.net/10400.22/6246 (accessed on 22 July 2024).
19. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6999–7019. [CrossRef]
20. Goswami, T.; Javaji, S.R. CNN Model for American Sign Language Recognition. In *Proceedings of the ICCCE 2020*; Kumar, A., Mozar, S., Eds.; Springer: Singapore, 2021; pp. 55–61. [CrossRef]
21. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255. [CrossRef]
22. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A Comprehensive Survey on Transfer Learning. *Proc. IEEE* **2021**, *109*, 43–76. [CrossRef]
23. TecPerson. Sign Language MNIST: Drop-In Replacement for MNIST for Hand Gesture Recognition Tasks. 2017. Available online: https://www.kaggle.com/datasets/datamunge/sign-language-mnist (accessed on 22 July 2024).
24. Lugaresi, C.; Tang, J.; Nash, H.; McClanahan, C.; Uboweja, E.; Hays, M.; Zhang, F.; Chang, C.L.; Yong, M.G.; Lee, J.; et al. MediaPipe: A Framework for Building Perception Pipelines. *arXiv* **2019**, arXiv:1906.08172.
25. Google AI. MediaPipe Gesture Recognizer. 2025. Available online: https://ai.google.dev/edge/mediapipe/solutions/vision/gesture_recognizer (accessed on 22 July 2024).
26. Kruse, R.; Mostaghim, S.; Borgelt, C.; Braune, C.; Steinbrecher, M. Multi-layer Perceptrons. In *Computational Intelligence: A Methodological Introduction*; Springer International Publishing: Cham, Switzerland, 2022; pp. 53–124. [CrossRef]
27. Takahashi, S. Hand Gesture Recognition Using MediaPipe. 2023. Available online: https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe (accessed on 22 July 2024).
28. Zhang, F.; Bazarevsky, V.; Vakunov, A.; Tkachenka, A.; Sung, G.; Chang, C.L.; Grundmann, M. MediaPipe Hands: On-device Real-time Hand Tracking. *arXiv* **2020**, arXiv:2006.10214.

29. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
30. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
31. He, H.; Garcia, E.A. Learning from Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284. [CrossRef]
32. Buda, M.; Maki, A.; Mazurowski, M.A. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Netw.* **2018**, *106*, 249–259. [CrossRef]
33. Powers, D.M.W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
34. Fahey, O.; Lexset. Synthetic ASL Alphabet. 2022. Available online: https://www.kaggle.com/datasets/lexset/synthetic-asl-alphabet/ (accessed on 22 July 2024).
35. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, 18–23 June 2018; pp. 4510–4520. [CrossRef]
36. Google. Flutter MediaPipe. 2024. Available online: https://github.com/google/flutter-mediapipe (accessed on 22 July 2024).