


Article

Enhancing Arabic Sign Language Interpretation: Leveraging Convolutional Neural Networks and Transfer Learning

Saad Al Ahmadi , Farah Muhammad * and Haya Al Dawsari

Department of Computer Science, College of Computer & Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia; salahmadi@ksu.edu.sa (S.A.A.); hs.aldawsari@psau.edu.sa (H.A.D.)

* Correspondence: fnazar@ieee.org

Abstract: In a world essentializing communication for human connection, the deaf community encounters distinct barriers. Sign language, their main communication method is rich in hand gestures but not widely understood outside their community, necessitating interpreters. The existing solutions for sign language recognition depend on extensive datasets for model training, risking overfitting with complex models. The scarcity of details on dataset sizes and model specifics in studies complicates the scalability and verification of these technologies. Furthermore, the omission of precise accuracy metrics in some research leaves the effectiveness of gesture recognition by these models in question. The key phases of this study are Data collection, Data preprocessing, Feature extraction using CNN and finally transfer learning-based classification. The purpose of utilizing CNN and transfer learning is to tap into pre-trained neural networks for optimizing performance on new, related tasks by reusing learned patterns, thus accelerating development and improving accuracy. Data preprocessing further involves resizing of images, normalization, standardization, color space conversion, augmentation and noise reduction. This phase is capable enough to prune the image dataset by improving the efficiency of the classifier. In the subsequent phase, feature extraction has been performed that includes the convolution layer, feature mapping, pooling layer and dropout layer to obtain refined features from the images. These refined features are used for classification using ResNet. Three different datasets are utilized for the assessment of proposed model. The ASL-DS-I Dataset includes a total of 5832 images of hand gestures whereas, ASL-DS-II contains 54,049 images and ASL-DS-III dataset includes 7857 images adopted from specified web links. The obtained results have been evaluated by using standard metrics including ROC curve, Precision, Recall and F-measure. Meticulous experimental analysis and comparison with three standard baseline methods demonstrated that the proposed model gives an impressive recognition accuracy of 96.25%, 95.85% and 97.02% on ASL-DS-I, ASL-DS-II and ASL-DS-III, respectively.

Keywords: Arabic Sign Language; CNN; transfer learning; ResNet-152**MSC:** 68U10

Citation: Al Ahmadi, S.; Muhammad, F.; Al Dawsari, H. Enhancing Arabic Sign Language Interpretation: Leveraging Convolutional Neural Networks and Transfer Learning. *Mathematics* **2024**, *12*, 823. <https://doi.org/10.3390/math12060823>

Academic Editors: Xinsong Yang and Alessandro Niccolai

Received: 16 January 2024

Revised: 22 February 2024

Accepted: 6 March 2024

Published: 11 March 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Globally, sign language is the primary mode of communication for over 70 million individuals. The development of an automated interpretation system holds the potential to greatly improve interactions between sign language users and those who are not proficient in it. Sign language encompasses not just hand movements but also integrates facial expressions, eye movements, and lip patterns to effectively convey messages. This form of communication is crucial for individuals who are deaf or have hearing difficulties, as it frequently represents their primary means of daily communication.

The World Health Organization reports that approximately 5% of the world's population experiences hearing loss [1]. This figure represents over 460 million individuals globally, including 34 million children [2]. Projections indicate that by 2050, the number

of people with hearing loss might increase to over 900 million. Additionally, noise exposure and other factors are putting around 1.1 billion young people at risk of hearing impairment [3]. The global economic burden of unaddressed hearing loss is estimated at 750 billion US dollars. Hearing loss can range from mild to profound, and those with severe to profound hearing loss often face significant communication challenges.

The deaf community primarily uses sign language, a gesture-based method of communication. However, this creates a divide, as most of the hearing population does not understand sign language [4]. There are approximately 200 different sign languages worldwide, each distinct in its way, just like spoken languages. This diversity further emphasizes the need for effective translation and interpretation solutions to bridge the communication gap between deaf and hearing individuals.

Sign language serves as a crucial communication method for the deaf community, distinguishing itself from other natural languages by primarily utilizing physical movements for conveying messages [5]. These movements, known as gestures or signs, encompass a variety of actions including hand and finger gestures, head nods, shoulder movements, and expressive facial features. The proposed work aims to facilitate interaction within the deaf community as well as between deaf individuals and those who can hear.

In sign language, each gesture or sign can represent a distinct letter, word, or even an emotion, similar to the way spoken languages use words to form sentences [6]. When a person who is deaf or has hearing difficulties wishes to convey a message, they employ these signs. The arrangement and combination of these signs create phrases and sentences, showcasing that sign language is not just a collection of isolated gestures but a fully fledged natural language with its own grammar and syntax [7]. This structured linguistic system allows for complex and nuanced communication, making it an essential tool for many in the deaf community for everyday interactions and expression.

The combination of computer vision and deep learning has driven remarkable progress in creating digital assistance systems, transforming sectors like agriculture and communication [8]. A key area of impact is sign language recognition, synthesis, and translation, which helps to bridge the communication divide between deaf individuals and the wider community. Static signs, which are characterized by distinct hand shapes and orientations without movement [9], are often used in finger spelling alphabet letters and numbers in various sign languages. The identification of these static signs is enabled by CNN models, which are adept at detecting and interpreting the stationary positions of hands and fingers.

Moreover, hand segmentation techniques, which are crucial for isolating the hand gestures from the background, employ sophisticated architectures. These techniques enhance the model's ability to focus on the relevant features of hand gestures [10]. Overall, the synergy of computer vision and deep learning in these systems not only facilitates efficient recognition of sign language but also paves the way for continuous innovation and enhancement in assistive technologies, significantly improving the quality of communication and interaction for deaf individuals.

In this research, the Arabic Alphabets Sign Language Dataset (ArASL2018) was utilized, which is a specialized dataset comprising images that depict each specific sign corresponding to the letters of the Arabic alphabet. This dataset forms an essential foundation for training and evaluating the proposed model's ability to recognize sign language. The experimental analysis conducted using the model demonstrated its efficacy in sign language recognition. The results were notably impressive, with the model achieving a recognition accuracy of 94.46%. This high level of accuracy is indicative of the model's robustness and its capability to accurately interpret the sign language gestures represented in the dataset.

Furthermore, a comparative analysis with previous studies was conducted to benchmark the performance of the proposed model. This comparison revealed that the proposed model not only performed well but actually outperformed all the previous studies in terms of recognition accuracy. This achievement highlights the advancements made in the field of sign language recognition and underscores the effectiveness of the methodologies and

technologies employed in the current model. Such a high degree of accuracy in recognizing Arabic sign language gestures is a significant milestone. It not only demonstrates the potential of deep learning and computer vision in assisting communication for the deaf community but also sets a new standard for future research and development in this field.

Research Contributions

The key contributions of the proposed research are as follows:

- By outperforming previous studies in recognition accuracy, the methodology establishes a new benchmark for future research in the field. This sets a higher standard for the development of sign language recognition systems and opens up new avenues for further improvements and innovations
- The proposed methodology demonstrates the successful application of advanced deep learning techniques in the domain of sign language recognition. This contribution is significant in showcasing the potential of AI in bridging communication gaps for the deaf and hard of hearing.
- One of the most notable contributions is the achievement of a high recognition accuracy of 94.46%. This level of precision in correctly identifying the signs for each letter of the Arabic alphabet represents a substantial improvement over previous methods.

The structure of the rest of this paper is organized as follows: Section 2 provides an extensive review of current techniques in the field. Section 3 delves into the fundamental methodology of Arabic Sign Language Recognition (ARSL), detailing its core principles. The implementation of the research and its simulation are thoroughly examined in Section 4. Section 5 concludes by discussing the findings of the study and suggesting possible avenues for further research.

2. Literature Review

This section focuses on existing models specifically designed for Arabic Sign Language recognition. These models are grounded in the realm of deep learning technology and utilize Deep Convolutional Neural Networks (CNNs). Emphasizing real-time hand gesture recognition, they aim to efficiently translate sign language, bridging communication gaps and enhancing accessibility for diverse communities. The exploration of these models sheds light on the advancements in technology that are making sign language translation more accessible and accurate.

Boutania et al. [11] focused on the development of an approach aimed at recognizing Arabic letters. The proposed system was crafted using a novel image pre-processing technique to accurately pinpoint the hand's position, coupled with a proposed architecture for a CNN utilizing depth data. The primary aim of their work was to facilitate smoother interactions between individuals with hearing impairments and the general population. Their system was designed to automatically detect and identify hand-signed letters from the Arabic alphabet based on user input. Their proposed model achieved an impressive recognition accuracy of 97.07% for Arabic sign language. For the validation of the effectiveness of their system, they also conducted a comparative study, which showed that their work surpassed the accuracy of similar studies on the same dataset, particularly in recognizing static signs.

Qanita et al. [12] introduced an inception V3 model, aimed at automatically and precisely identifying Arabic Sign Language characters. Utilizing a dataset of 54,049 ArSL letter images, their research uncovered that the InceptionV3 model stands out among other pretrained models. Their exceptional results not only demonstrated the superior ability of InceptionV3 in accurately recognizing Arabic characters but also highlighted its resilience against overfitting. The success of InceptionV3 in their study paved the way for its potential application in future Arabic Sign Language recognition research, marking a significant advancement in this specialized area of study.

Herbaz et al. [13] introduced two unique datasets: the first consists of 54,049 images from the ArSL-2018 alphabet, captured by more than 40 individuals, and the second is a

personal dataset containing 15,200 images. These images underwent processing with the VGGNet model and ResNet50, which included classification, normalization, and detection techniques. The VGGNet layers, initially pre-trained on extensive alphabet sign language datasets, were further optimized, while the ResNet50 classifier was trained over 40 epochs with subsequent fine-tuning. This approach achieved accuracy levels that surpassed those in other studies, demonstrating the effectiveness of these methods. Table 1 presents a summary of some related studies.

To bridge the gap and alleviate the isolation of the hearing-impaired community, the development of automatic sign language interpreters is becoming increasingly essential [14]. Their work proposed a hybrid approach combining modified Convolutional Neural Networks (CNNs) with machine learning classifiers. This innovative method recognized both Arabic sign language alphabets and symbolic signs (words). A new dataset comprising images from existing datasets, video frames, and camera captures were also designed. These images undergo preprocessing and feature extraction using Linear Discriminant Analysis. The extracted features then feed into a one-dimensional convolutional neural network, which utilized one of three classifiers—Decision Trees, Naive Bayes and Random Forest—replacing the conventional neural network approach. The performance of the proposed algorithm was rigorously tested under various challenges, demonstrating exceptionally high accuracy rates of up to 99.9% in recognizing both alphabets and words. Moreover, it showed great potential for efficient real-time application.

Table 1. Comparative analysis of the relevant literature based on their strengths and weakness.

Ref#	Approach	Dataset Size	Accuracy Achieved	Strengths	Weakness
[15]	MobileNet, Xception	54,049 images	99.9% (alphabet and words)	Assessment of Diverse Pretrained Models and Vision Transformers; Outstanding Performance in Arabic Sign Language Classification	Dependence on large dataset size; potential overfitting with complex models
[16]	Deep Learning-Based Classification with Transfer Learning on 12 Image Recognition Models	Aplhabet Arsl 19	93.7% (translation accuracy)	Focuses on translation accuracy; faster than traditional models	Limited information on dataset size and specific model details; potential scalability issues
[17]	Convolutional Neural Networks with EfficientnetB1 for Arabic Sign Language Interpretation	16,192 images	99% accuracy, 97.9% validation	EfficientnetB1 scaling for improved CNN performance; high accuracy in ArSL classification	Reliance on a specific CNN architecture; may not generalize well to other sign languages
[18]	Static Hand Gestures Recognition for ArSL using CNN	Arsl 2018	97.42%	Effective static gesture recognition for ArSL with high accuracy	Limited to static gestures; may not effectively handle dynamic or complex signs
[19]	Dynamic Model for ArSL Recognition using CNN	Arsl 2019	High (not quantified)	Dynamic recognition approach; demonstrates effectiveness across multiple gestures	Lack of quantified accuracy figures; may not be comprehensive in gesture coverage
[20]	Deer Hunting Optimization based Classification	Self Made 1423 images	92.88%	Combines DenseNet169 with MLP and DHO algorithm for enhanced gesture classify	Specific to gesture classification; may not account for contextual nuances in sign language

3. Materials and Methods

In this part, the structure of the ArSL-CNN model is described, tailored specifically for Arabic Sign Language classification. The core components of the proposed model include Data Preprocessing, CNN-based Feature Extraction, and Classification via Transfer Learning, as demonstrated in Figure 1. Each of these steps plays a critical role in ensuring the effective and accurate recognition of Arabic Sign Language gestures, leveraging the strengths of convolutional neural networks and transfer learning techniques.

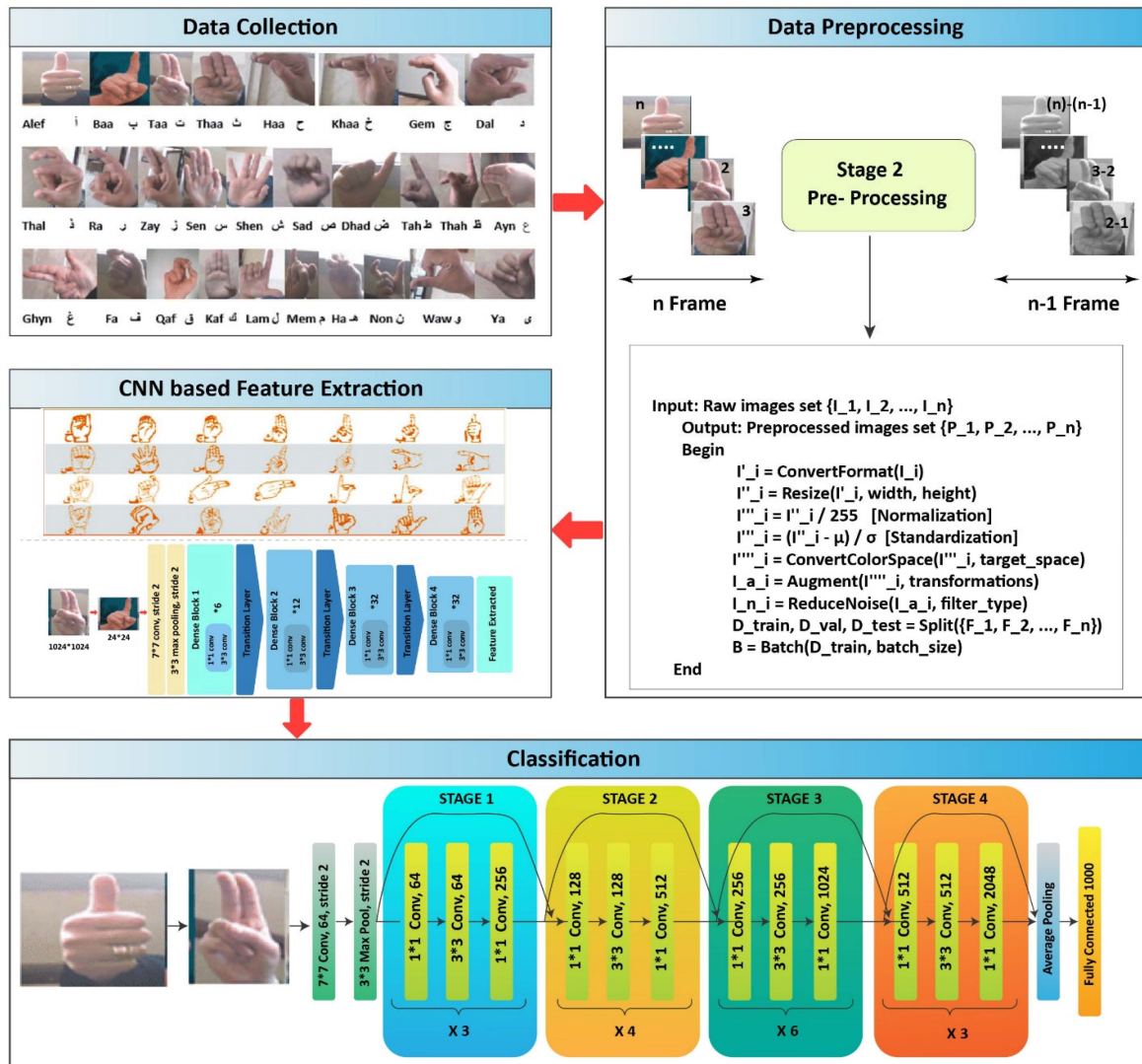


Figure 1. Schematic architecture of proposed sign language recognition model.

3.1. Data Collection

This research utilizes three datasets majorly composed of images of different signs and alphabets. To enhance the study’s outcomes, a large dataset was selected. The images in the dataset vary in terms of lighting, angles, timings, and backgrounds, and were initially captured in full-color (RGB) mode. Due to their varying sizes, a preprocessing step was implemented to standardize the images for classification. This involved resizing them to 64×64 pixels and converting them to grayscale with a pixel intensity range of 0–255. Each image in the dataset is meticulously labeled according to its corresponding sign language letter. The composition of the dataset for each letter is detailed in Figure 2.

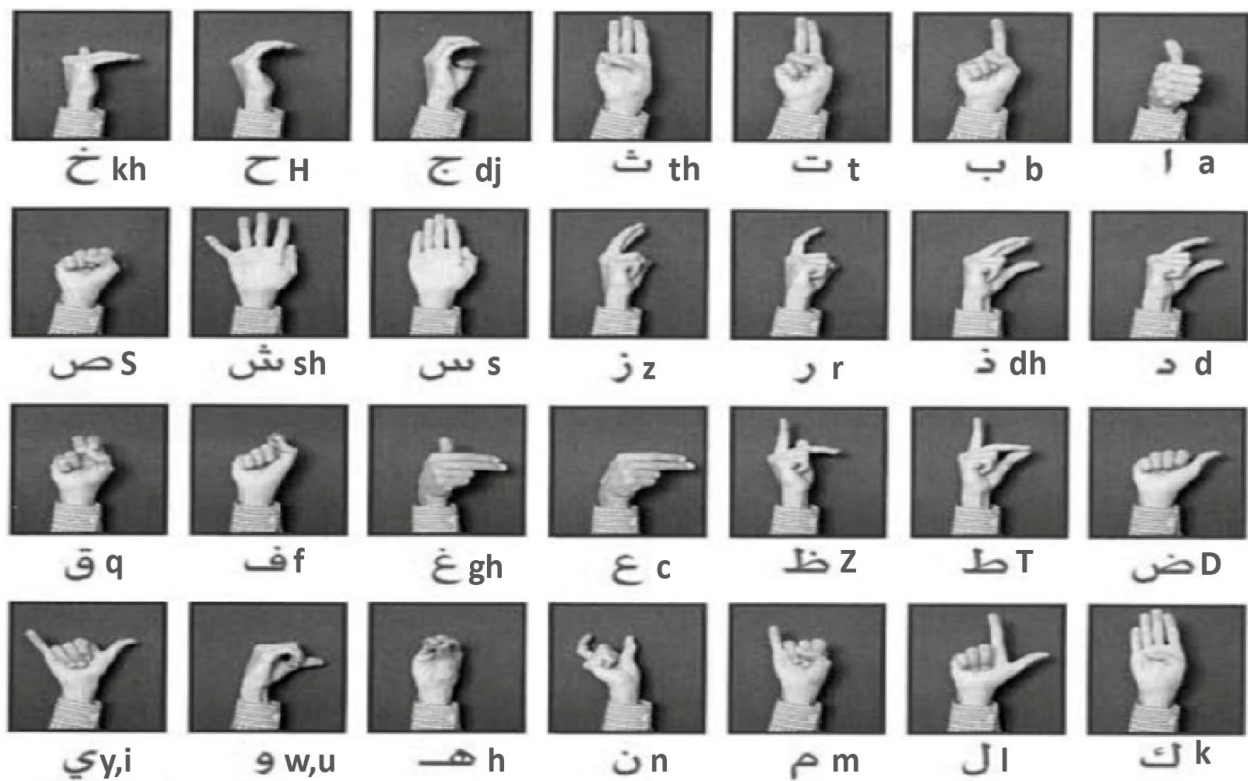


Figure 2. The Arsl2018 Alphabet dataset with Correspondence Sign.

3.2. Data Preprocessing

Data preprocessing involves a sequence of morphological procedures designed to eliminate noise from the dataset. The ArSL2018 dataset, which includes images of sign language gestures, contains images with varying sizes and levels of lighting [21]. To make sure the data is uniform and free of impurities, image preprocessing methods are essential. Initially, all images of sign language gestures are transformed into 64×64 pixel greyscale images. This step is key for efficient real-time classification. By converting images to greyscale, which uses a single channel instead of the typical three RGB channels, the complexity of the first convolutional layer is reduced by half. This reduction in parameters leads to decreased computational time.

This Algorithm 1 outlines a process for preprocessing a set of raw images for use in machine learning tasks. It starts by converting the images to a consistent format and size, then normalizes and standardizes their pixel values to ensure they're within a specific range and distribution. The color space of the images is also converted to a target specification. Image augmentation techniques are applied to introduce variability, followed by noise reduction to improve image quality. Finally, the preprocessed images are split into training, validation, and test datasets, and the training set is batched for efficient processing. This sequence of steps prepares the images for effective use in training machine learning models by ensuring consistency, enhancing model robustness, and facilitating faster convergence.

To further enhance computation efficiency and accelerate the training process, all images undergo normalization to scale pixel values from 0 to 1. Standardization is also applied by subtracting the mean and scaling to unit variance [22]. For generating training and testing sets, images are randomly chosen from the dataset, with 70–30% division is used for training and testing the data, respectively, with 20% of the training dataset set aside for validation purposes. As depicted in Figure 3, there is an uneven distribution of classes within the dataset, leading to the implementation of diverse resampling methods to correct this imbalance in class representation.

Algorithm 1 Proposed Preprocessing Algorithm for Refining Image Dataset

Input: Raw images set $\{I_1, I_2, \dots, I_n\}$
Output: Preprocessed images set $\{P_1, P_2, \dots, P_n\}$
Begin
 $I'_i = \text{ConvertFormat}(I_i)$
 $I''_i = \text{Resize}(I'_i, \text{width}, \text{height})$
 $I'''_i = I''_i / 255$ [Normalization]
 $I''''_i = (I'_i - \mu) / \sigma$ [Standardization]
 $I''''_i = \text{ConvertColorSpace}(I''''_i, \text{target_space})$
 $I_{a_i} = \text{Augment}(I''''_i, \text{transformations})$
 $I_{n_i} = \text{ReduceNoise}(I_{a_i}, \text{filter_type})$
 $D_{\text{train}}, D_{\text{val}}, D_{\text{test}} = \text{Split}(\{F_1, F_2, \dots, F_n\})$
 $B = \text{Batch}(D_{\text{train}}, \text{batch_size})$
End

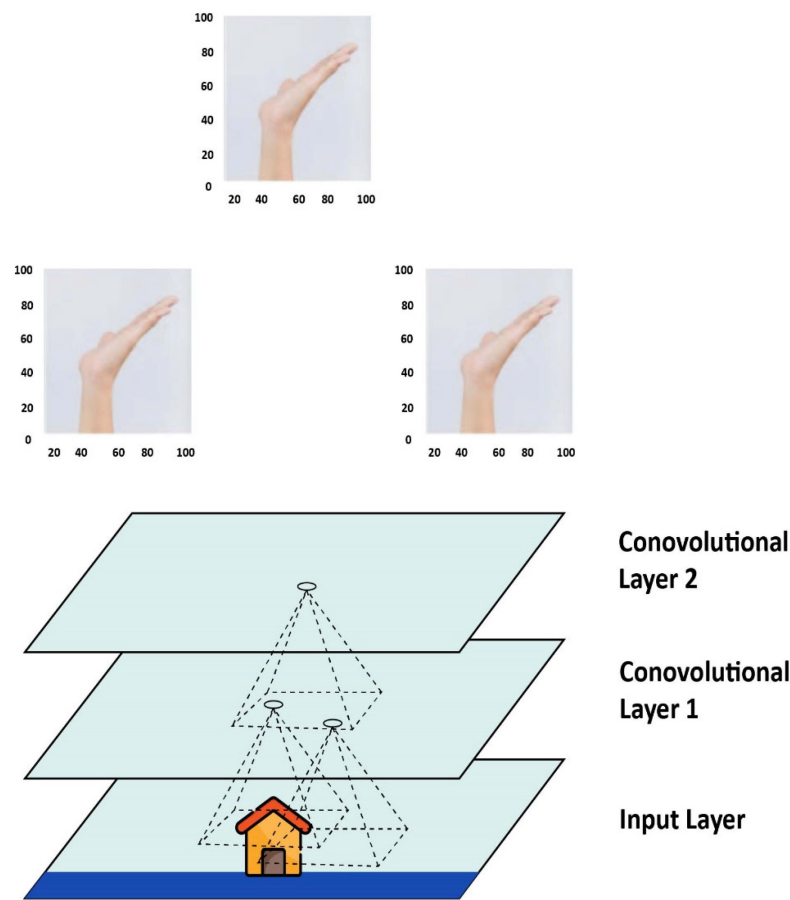


Figure 3. Rectangular Local Receptive Field Layers in CNNs.

3.3. CNN Based Feature Extraction

The use of a multilayer architecture alone proves to be inadequate for image recognition tasks, especially considering the complexity and size of the networks involved. For instance, a relatively simple 100×100 pixel image results in 10,000 neurons. When these neurons are connected to a layer of 1000 neurons, the network ends up having an overwhelming 10 million connections right at the first hidden layer. Managing such a vast number of connections is not only challenging but also computationally inefficient [23,24]. While multilayer models may perform acceptably with small images that possess fewer features, they struggle with larger or more complex datasets. Conversely, Convolutional Neural Networks (CNNs) offer a more feasible solution for such tasks. CNNs utilize specialized layers designed to reduce the number of connections, thereby streamlining the

network for more efficient classification without compromising on the quality of feature detection and extraction.

3.3.1. The Convolution Layer

In CNN, an essential component of their architecture is the specialized approach to feature extraction [25]. Unlike in traditional neural networks where neurons are fully connected, in CNNs, neurons in the first convolution layer connect only to a specific subset within their receptive field, focusing on small, localized regions of the input. This selective connectivity continues in subsequent layers, where each neuron in a layer like the second convolution layer connects to a group of neurons within a defined area in the preceding layer. In CNNs, the initial layers usually detect fundamental features such as edges and textures. In contrast, the more advanced layers are capable of identifying complex and abstract characteristics. This structure makes CNNs particularly efficient for tasks that require detailed visual data analysis, like image recognition.

The array of neurons used for scanning the input image is known as the kernel or filter. This kernel moves across the input layer to perform convolution operations, but this process does not inherently ensure that the center of the kernel aligns with the outermost elements of the input layer. To address this, padding, specifically Zero Padding, is employed [26]. Zero Padding involves adding layers of zeros around the input image, allowing the kernel to fully cover the border areas of the input without missing any part. This technique ensures that every element of the input layer, including the outermost elements, can be processed by the center of the kernel. A stride is essentially the number of pixels by which the kernel shifts as it convolves around the input. This parameter can be adjusted to control how broadly the kernel moves across the input layer.

3.3.2. Feature Mapping

The convolutional layer is often visualized in 3D, comprising multiple feature maps. Each feature map acts as a distinct filter, designed to identify specific features within the input image, such as vertical or horizontal lines. This multi-faceted approach to feature detection is depicted in Figure 4, where various feature maps within a convolutional layer are illustrated, each focusing on extracting different types of features from the input.

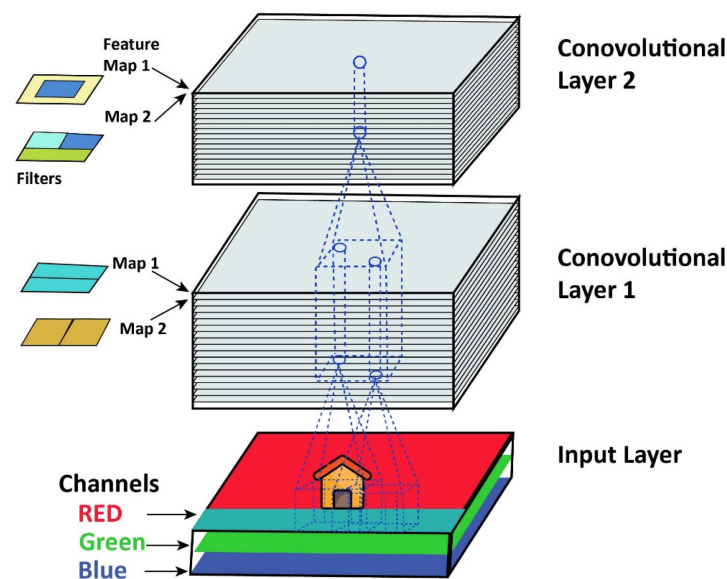


Figure 4. Convolutional Layer with Multiple Feature Maps for Enhanced Feature Extraction.

The equation that shows the convolutional layer computes the following output:

$$Z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k=0}^{f_n-1} X_{i,j,k} \cdot W_{u,v,k} \tag{1}$$

where

$Z_{i,j,k}$: the output of the neurons located at row i and column j in the k th feature map of the convolutional layer;

$X_{i,j,k}$: the output of i th neuron in layer $L - 1$;

b_k : the bias term in layer L ;

$W_{u,v,k}$: the connection weights.

3.3.3. Pooling Layer

The pooling layer in a CNN serves to reduce computational load by downsizing the inputs [27]. Similar to the convolutional layer, it connects to the outputs of the previous layer but only partially, focusing on small rectangular receptive fields. Pooling comes in two main forms: Max pooling and Mean pooling. Max pooling, the more commonly used type, selects the maximum value from the receptive field in the feature map, emphasizing the most prominent features. Mean pooling, on the other hand, calculates the average of all elements within the receptive field, providing a more generalized feature representation. Figure 5 demonstrates the use of both Max and Mean pooling methods. The figure clearly shows that applying a 2×2 pooling kernel with a stride of 2 diminishes the image's dimensions to one-fourth of its initial size. This decrease in size substantially reduces computational demands, memory needs, and parameter count, thereby making the process of extracting features from the image more efficient.

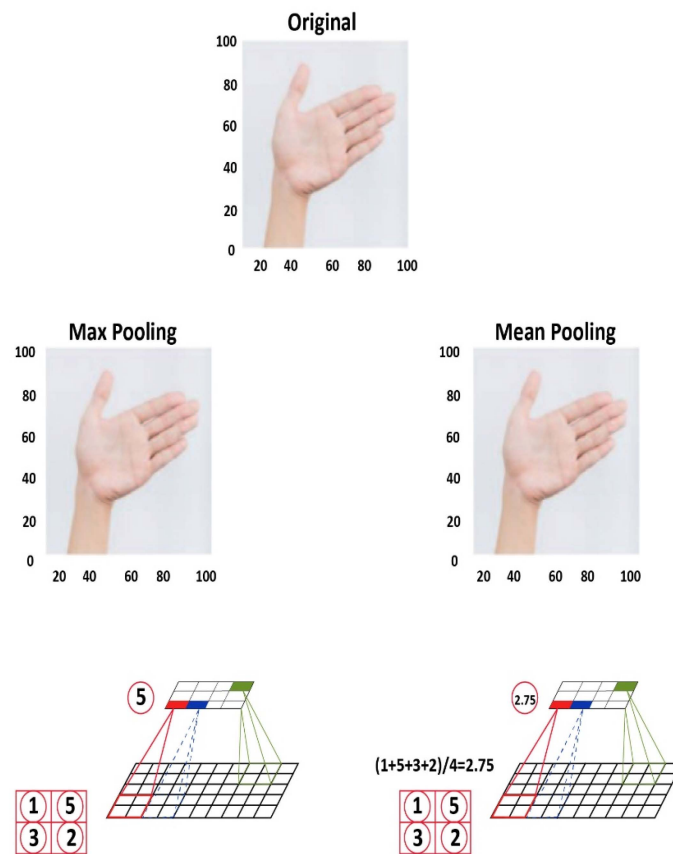


Figure 5. Max pooling and mean pooling layer of CNN for feature extraction.

3.3.4. Dropout Layer

A neural network, with its vast array of parameters, provides considerable flexibility but also becomes susceptible to overfitting. Overfitting occurs when a model learns too much from the training data, including its noise and irregularities, leading to a decrease in its performance on new, unseen data. To counteract overfitting, one effective approach is to

implement early stopping in the training process. Performance. However, with unlimited computations, early stopping can be overly aggressive and time-consuming.

This method entails tracking the network's performance on a separate validation set and halting the training phase once there is a noticeable decline in performance on this dataset. The network thus retains the parameters that yielded the best validation. An alternative and often more effective approach to prevent overfitting is the implementation of regularizes, with dropout being one of the most commonly used techniques. Dropout works by randomly 'dropping out' neurons and their incoming and outgoing connections from the network during the training phase, as depicted in Figure 6. This process is random, or in other words, each neuron is dropped with a probability defined by the dropout rate (ρ), typically set between 0.4 and 0.5 in CNNs. It is important to note that during the testing phase, dropout is not applied, and all neurons are active.

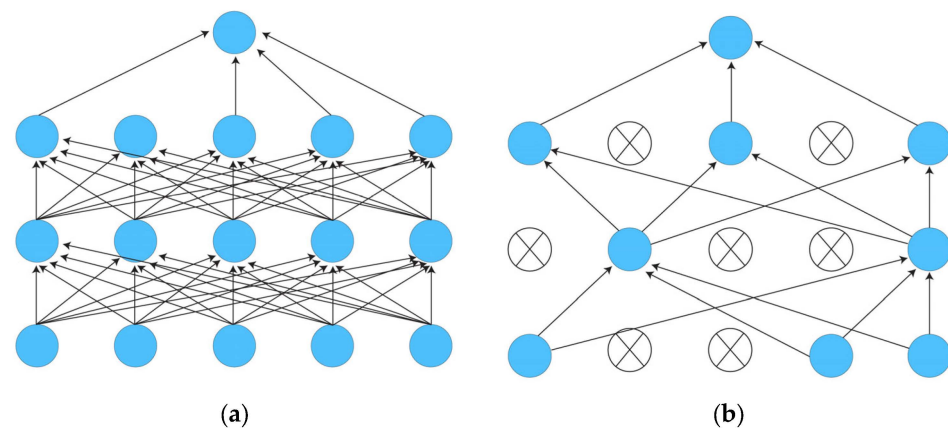


Figure 6. (a) Network without dropout; (b) network with dropout.

To illustrate further, consider a dropout rate (P) of 0.5. During testing, the neurons are effectively receiving double the input signal compared to the training phase, where only half of the neurons were active on average. This discrepancy can lead to performance issues. To compensate, the input signal to each neuron in the testing phase is scaled down, typically by multiplying it by the dropout rate (in this case, 0.5). This scaling ensures consistency in the network's behavior between the training and testing phases. In order to avoid overfitting, cross-validation was used, which involves dividing the dataset into multiple parts, using some for training and others for validation. By repeatedly training and evaluating the model on different subsets, you can ensure that it generalizes well to unseen data. Algorithm 2 demonstrates the operation of a neural network without dropout and Algorithm 3 demonstrates the operation of a neural network with dropout.

Algorithm 2 Train Neural Network without Dropout

Input: Training data (X, Y), number of epochs, learning rate

Output: Trained model parameters

```

1: Initialize network weights and biases
2: for each epoch in number of epochs do
3:   for each (x, y) in training data do
4:     // Forward Propagation
5:     A[0] = x
6:     for each layer L do
7:       Z[L] = W[L] * A[L - 1] + b[L]
8:       A[L] = activation(Z[L])
9:     end for
10:
11:   // Compute Loss

```

Algorithm 2 Cont.

```

12:     Loss = compute_loss(A[last layer], y)
13:
14:     // Backward Propagation
15:     for each layer L in reverse do
16:         dZ[L] = derivative of loss with respect to Z[L]
17:         dW[L] = dZ[L] * A[L - 1].T
18:         db[L] = sum(dZ[L], axis = 1, keepdims = True)
19:         dA[L - 1] = W[L].T * dZ[L]
20:     end for
21:
22:     // Update Parameters
23:     for each layer L do
24:         W[L] = W[L] - learning rate * dW[L]
25:         b[L] = b[L] - learning rate * db[L]
26:     end for
27: end for
28:
29: // Optional: Early Stopping based on validation performance
30: if validation performance decreases then
31:     break
32: end if
33: end for
34:
35: return W, b

```

Algorithm 3 Train Neural Network with Dropout**Input:** Training data (X, Y), number of epochs, learning rate, dropout rate ρ **Output:** Trained model parameters

```

1: Initialize network weights and biases
2: for each epoch in number of epochs do
3:     for each (x, y) in training data do
4:         // Forward Propagation with Dropout
5:         A[0] = x
6:         for each layer L do
7:             Z[L] = W[L] * A[L - 1] + b[L]
8:             A[L] = activation(Z[L])
9:             if L < last layer then
10:                D[L] = (random numbers <  $\rho$ )
11:                A[L] = A[L] * D[L]
12:                A[L] = A[L] /  $\rho$ 
13:            end if
14:        end for
15:
16:        // Compute Loss
17:        Loss = compute_loss(A[last layer], y)
18:
19:        // Backward Propagation with Dropout
20:        for each layer L in reverse do
21:            if L < last layer then
22:                dA[L] = dA[L] * D[L]
23:                dA[L] = dA[L] /  $\rho$ 
24:            end if
25:            dZ[L] = derivative of loss with respect to Z[L]
26:            dW[L] = dZ[L] * A[L - 1].T

```

Algorithm 3 *Cont.*

```

27:         db[L] = sum(dZ[L], axis = 1, keepdims = True)
28:         dA[L - 1] = W[L].T * dZ[L]
29:     end for
30:
31:     // Update Parameters
32:     for each layer L do
33:         W[L] = W[L] - learning rate * dW[L]
34:         b[L] = b[L] - learning rate * db[L]
35:     end for
36: end for
37:
38: // Optional: Early Stopping based on validation performance
39: if validation performance decreases then
40:     break
41: end if
42: end for
43:
44: return W, b

```

3.4. Classification Using Transfer Learning

Transfer learning is a powerful technique in machine learning, especially for image classification tasks. It involves using a pre-trained model, which has been trained on a large and general dataset, as the starting point for training on a new, usually smaller, dataset for a specific task. The effectiveness of transfer learning lies in leveraging the learned features and patterns from the extensive initial training, which can significantly improve the performance on the new task, especially when the available data for the new task is limited. Transfer learning bridges the gap between the availability of large-scale annotated datasets and the need for specialized image classification tasks. It is a go-to strategy in many real-world applications, providing a head start in model training and boosting performance, especially when computational resources or labeled data are limited.

ResNet, short for Residual Network, was introduced by researchers from Microsoft. It is particularly known for its ability to enable the training of very deep neural networks. It is widely used for transfer learning in various machine learning tasks, especially in image processing and computer vision. Transfer learning involves taking a model trained on one task and applying it to a different, but related task.

3.5. Classification Layer

The classification process starts with a pre-trained model, ResNet 152 which has been initially trained on a large dataset like ImageNet. This pre-training provides the model with a comprehensive understanding of various image features [28]. When this model is applied to a new classification task, the primary goal is to adjust the model so it can effectively categorize images according to the new task's requirements.

In fine-tuning for classification, the entire network, including the pre-trained layers, is trained (or retrained) on the new dataset. This is a departure from feature extraction, where the pre-trained layers are kept frozen. The fine-tuning process involves adjusting the weights across all layers of the network. These adjustments enable the model to adapt its previously learned feature detection capabilities to the specifics of the new classification task.

During fine-tuning, it is common to replace the final layer (or layers) of the pre-trained model with new ones that are tailored to the new task. For instance, if the original model was trained to classify 1000 different categories but the new task only involves 10, the final classification layer would be modified to output 10 categories instead of 1000. The ResNET model is shown in Figure 7.

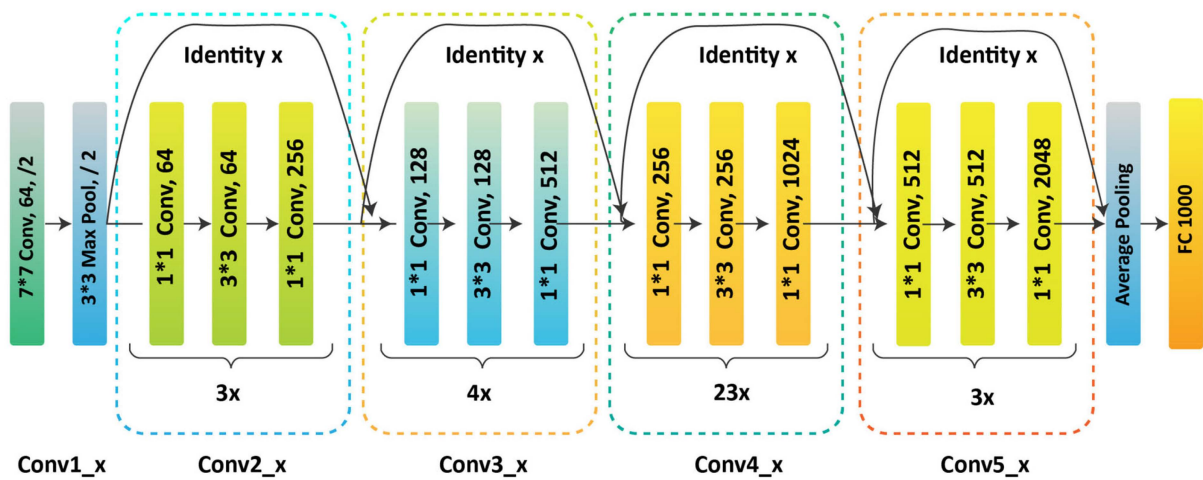


Figure 7. ResNet Model for classification of Arabic Sign Language.

Let us denote the function of the original ResNet model as $F(x; \theta)$, where x is the input image, and θ represents the weights of the model. The output of this function is a vector of probabilities across the original set of classes, say C , where C could be 1000 for ImageNet.

$$F(x; \theta) = [p_1 + p_1 + p_1, \dots, p_n] \tag{2}$$

The classification task includes N classes (where N is different from C); the original output layer of ResNet is replaced. Let ϕ denote the weights of the new output layer. The modified model function can be represented as

$$G(x; \theta, \phi). \tag{3}$$

During fine-tuning, we adjusted both θ and ϕ using the new dataset. The goal was to reduce the loss function L , which quantifies the discrepancy between the predicted outcomes and the true labels in the new dataset. The loss function could be cross-entropy for classification tasks.

$$\min_{\theta, \phi} L(G(x; \theta, \phi), y) \tag{4}$$

where y represents the actual label of the image x in the dataset.

Algorithm 4 outlines the fine-tuning process for a pre-trained ResNet model. It involves modifying the output layer for the new number of classes (N), and then iteratively training (both forward and backward passes) on the new dataset. The weights of the pre-trained layers (θ) and the new output layer (ϕ) are updated using their respective learning rates (α for θ , β for ϕ) based on the gradients of the loss function. The fine-tuning is performed over a specified number of epochs to adapt the model for the new classification task.

Algorithm 4 Fine-Tuning of Pre-Trained ResNet Model for Arabic Sign Language Recognition

Inputs:

- PreTrainedResNetModel: ResNet model pre-trained on ImageNet (weights θ)
- NewDataset: Dataset for the new classification task
- N : Number of classes in the new task
- α : Learning rate for θ
- β : Learning rate for ϕ
- Epochs: Number of training epochs

Procedure:

- 1: Modify PreTrainedResNetModel:
-

Algorithm 4 *Cont.*

Replace the output layer to have N nodes (weights φ)

```

2:  for each epoch in 1 to Epochs do
3:    for each batch (X, Y) in NewDataset do
4:      // Forward Pass
5:      PredictedOutput = G(X;  $\theta$ ,  $\varphi$ )
        // G: Modified ResNet function

6:      // Compute Loss
7:      Loss = L(G(X;  $\theta$ ,  $\varphi$ ), Y)
        // L: Loss function (e.g., cross-entropy)

8:      // Backward Pass and Update
9:      Compute gradients:  $\nabla\theta L$ ,  $\nabla\varphi L$ 
10:     Update  $\theta$ :  $\theta = \theta - \alpha * \nabla\theta L$ 
11:     Update  $\varphi$ :  $\varphi = \varphi - \beta * \nabla\varphi L$ 
12:   end for
13: end for

```

4. Experimental Results and Analysis

The present section contains an analysis of the experiment of this study which includes the description of performance metrics, baseline methods, dataset detail and the obtained results.

4.1. Dataset Description

Three different types of datasets were used for the analysis of the proposed model. In the ASL-DS-I dataset, which is an ArASL2018 dataset comprising 54,049 fully annotated images, representing 32 alphabets of Arabic sign language. These images were captured from 40 participants, who varied in age. Each image was in grayscale, formatted as 64×64 pixels JPG, and was taken using a smart camera. In ASL-DS-II more than 40 individuals contributed to compile 54,049 images as a new dataset of ArSL alphabets. Typically, the classification of alphabet deployments of the standard Arabic letters and digits ranged from 32 different hand signs and alphabets. The constituent class intervals were dispersed unevenly. The ASL-DS-III dataset consisted of total 7857 raw and fully labeled RGB images. The dataset collected data from more than 200 participants and different settings including lighting, different image orientations, the different background, different image size, and image resolution values. All images were supervised and filtered by the field experts with the confident level to have a better-quality dataset. The dataset description is shown in Table 2.

Table 2. Dataset description.

Dataset ID	Name	Size	Web Link
ASL-DS-I	Arabic Sign Language ArSL dataset	5832 images	https://shorturl.at/bjmtL , accessed on 10 January 2024
ASL-DS-II	Arabic Alphabets Sign Language Dataset (ArASL)	54,049 images	https://shorturl.at/vBLVY , accessed on 10 January 2024
ASL-DS-III	RGB Arabic Alphabets Sign Language Dataset	7857 images	https://shorturl.at/clyI6 , accessed on 12 January 2024

4.2. Experimental Enjoinment

With the help of the Tensor Flow and Keras libraries, Python was used in order to put the suggested model into action [23]. In order to carry out the experiment, a CPU system with a moderate price tag was used. For experiments, an NVIDIA GeForce RTX 3080, with 8 GB of VRAM was used. This offers a good balance between cost, performance, and memory size. A CPU with a speed of 2.60 GHz Intel Core-i5-3230 and 8 GB of data storage was also used. The training parameters are given in Table 3.

Table 3. Training Parameters.

#	Name	Web Link
1	Learning Rate	0.001
2	Batch Size	64
3	Number of Epochs	100

4.3. Performance Matrices

In order to assess the effectiveness of the proposed work, the performance matrices that are shown below were used.

- A confusion matrix is a 2 by 2 table with four outputs from a classifier that is produce as a result of building the model.
- The definition of accuracy is the amount of times that are truly classified or divided by the total times.

$$\text{Accuracy} = \frac{(t_p + t_n)}{(t_p + t_n + f_p + f_n)} \quad (5)$$

- Precision (P): precision prolongs upon the error extent over the indicators of either all the cases of the positive class, or all the cases that have been reckoned by the learning system to be positive.

$$\text{Precision} = \frac{(t_p)}{(t_p + f_p)} \quad (6)$$

- Recall (R): this show how many of the positive instances it has classified actually are positive.

$$\text{Recall} = \frac{(t_p)}{(t_p + f_n)} \quad (7)$$

- F-measure (F): this has been used to contrast algorithms, with the F1 score defined as the harmonic mean of sensitivity and precision.

$$\text{F-measure} = \frac{2 * P * S}{P + S} \quad (8)$$

- The ROC curve provides a visual representation of the ability of a binary classification model to discriminate between classes.

4.4. Baseline Methods

For the purpose of evaluating the effectiveness and quality of the work that is being suggested, the following baseline approaches have been used.

- Herbaz et al. [13] used two different models as the bases for this analysis, VGGNet and ResNet50. The authors trained and tested their algorithm both without and with fine-tuning enhancers. The results of recognizing the alphabets in ArSsl dataset were 89.05%, 89.99%, and 88.50% respectably.
- Latif et al. [29] presents Deep CNN architectures like AlexNet and ZFNet. A number of experiments were carried out and CNN architectural design parameters were changed in almost all the experimentations in order to achieve the optimal and best possible recognition rates. Their proposed Deep CNN architecture achieved an accuracy of 94.06%, as depicted in the experimental results.
- Aldhahri et al. [30] used a CNN to develop a model of classifying Arabic language alphabet signs to help reach the goal of communication. The outcome of the analysis showed an accuracy rate of 94.46%.

4.5. Results

The performance of the proposed model on different datasets is presented in Figure 8, using Accuracy, Specificity and Sensitivity, respectively. It was observed that the proposed model performed excellently on the ASL-DS-I Dataset with 94.56% precision, 93.25% recall, 96.25% accuracy and a 93.90% F-measure. Also, the ASL-DS-II dataset that was used to test the model was also a very excellent platform for this project as the model achieved 94.84% precision, 94.56% recall, a 94.70% F-measure and 95.85% accuracy. ASL-DS-III dataset were also obtained and used to test the model in order to see how well a dataset with mostly close in features images will be amenable and responsive to the project and the model returned 96.74% precision, 96.56% recall, 97.02% accuracy and 96.65% F-measure.

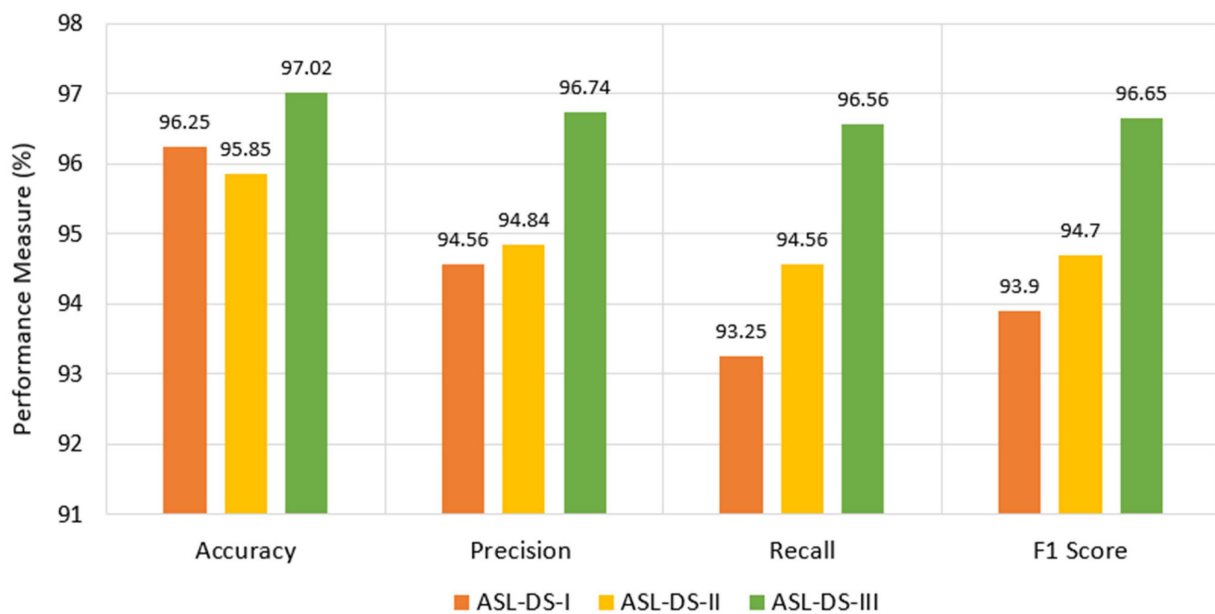


Figure 8. Experimental results in terms of accuracy, precision, recall, and F1 score on multiple datasets.

Figure 9 depicts the Receiver Operating Characteristic (ROC) curves that show the performance of the proposed approach. These curves measure the True Positive Rate (TPR), and the False Positive Rate (FPR) for the various datasets. One of the evaluation metrics for model performance on the various datasets is the Area Under the Curve (AUC) of the ROC curve. The model achieved an AUC of 0.81 for the ASL-DS-1 Dataset, 0.83 for the ASL-DS-II Dataset and 0.72 for the ASL-DS-III Dataset. The ROC curves for two of the datasets covered more than 80% of the area, while the third covered more than 72%, showing then that the model is performing well in differentiating between the positive and negative classes on the different dataset, thereby validating the performance of the model.

The first comparative analysis consisted of the performance test of proposed model with six different combination of standard deep learning models each one having feature extraction and sign language recognition stages. The configurations include Single Layer LSTM, LSTM-LSTM, Single Layer GRU, GRU-GRU, GRU-LSTM and LSTM-GRU.

Each model was evaluated for their performance with K-fold cross validation. Most notably, the LSTM-GRU model showed the highest accuracy among the other six type of configuration. For the development of the models were performed using Keras/TensorFlow libraries which have been widely utilized for their great flexibility and efficiency in deep learning model construction. Graphically, the results are shown in Figure 10.

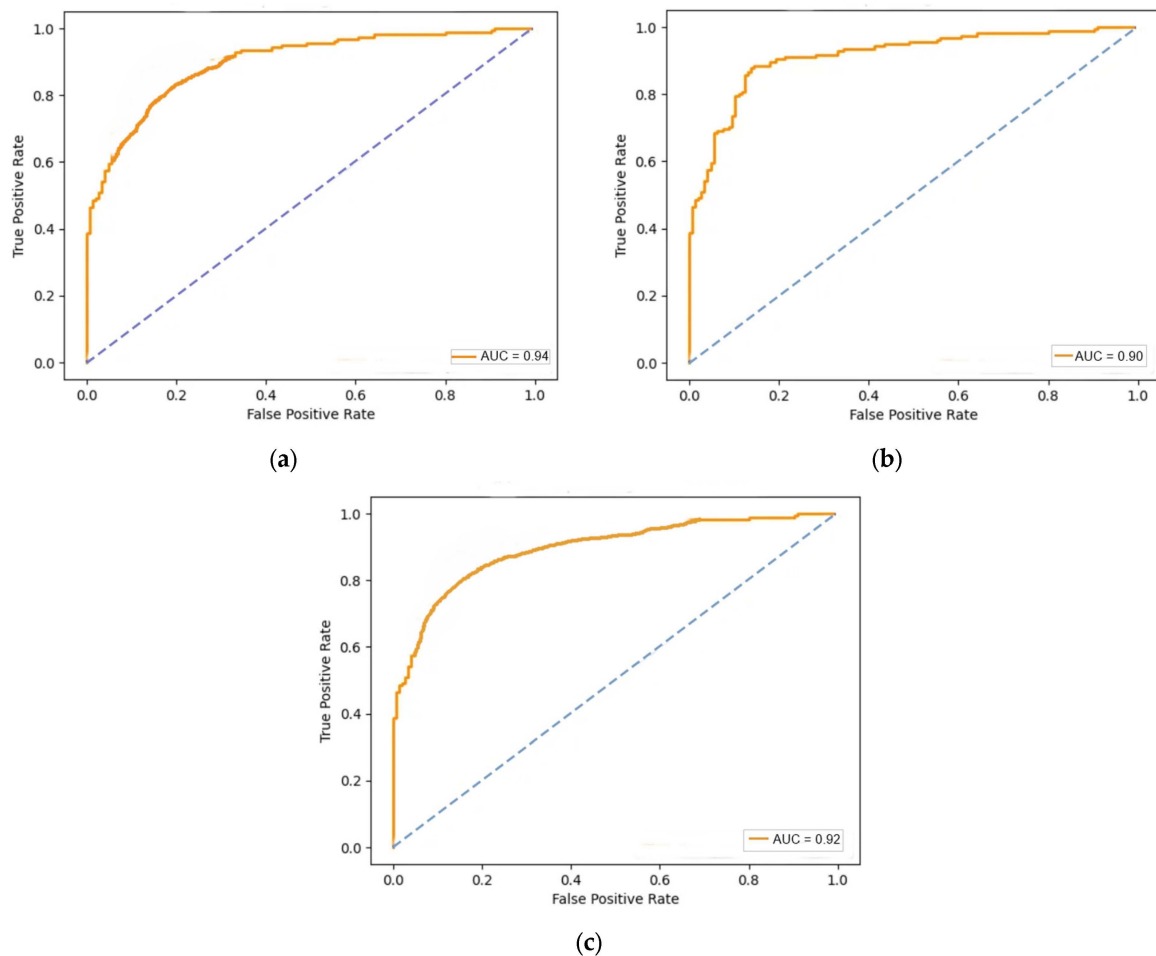
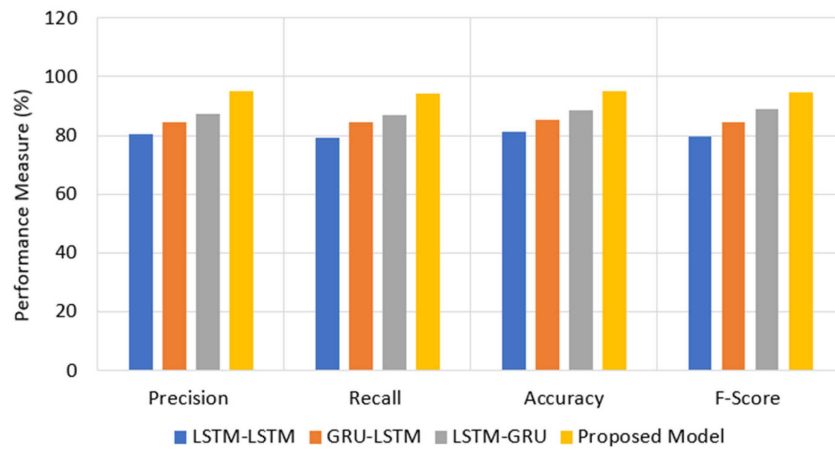


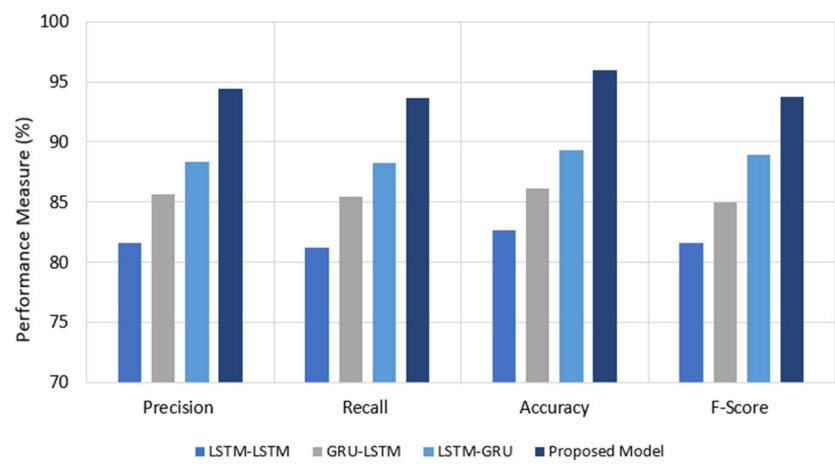
Figure 9. ROC curve-based analysis on multiple datasets for proposed sign language recognition model. (a) ASL-DS-I Dataset; (b) ASL-DS-II Dataset; (c) ASL-DS-III Dataset.

The main objective of these models is the discovery of sign language. It is likely that the performance results of the models will get better once we increase our number of samples per word and also the size of our datasets. The use of this method will give the models more varieties of training examples, resulting in the discovery of the sign language in a more effective and sought-after method.

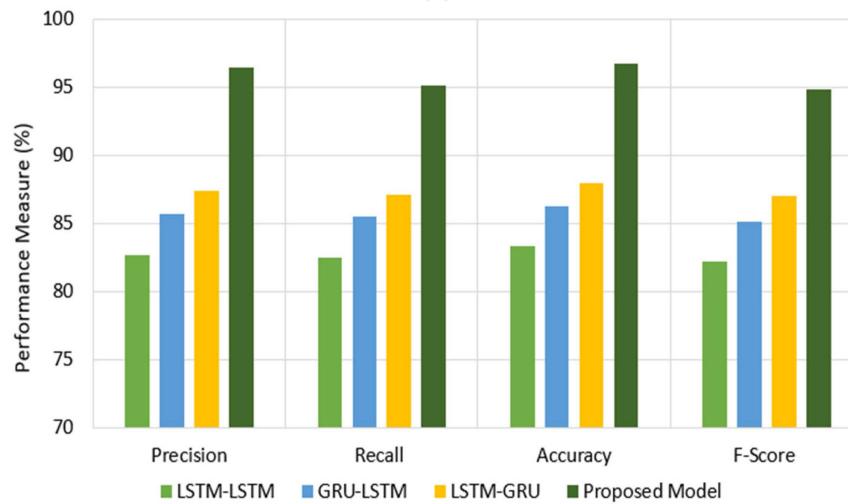
An additional experiment was conducted comparing the proposed model to the standard baseline models, as mentioned in Section 4.4. As can be seen in Figure 11, it was found that the proposed model achieved an accuracy of 96.37%, which outperformed the other existing models, Herbaz et al. [13], Latif et al. [29] and lastly Aldhahri et al. [30]. This experiment thus points out the competitiveness and the viability of the proposed model and hence it could be used in applications which demand the overall accuracy as the primary metric being operational, even though the methodologies underlying these models, as mentioned above are quite different.



(a)



(b)



(c)

Figure 10. Comparative analysis of proposed model with standard deep learning models. (a) ASL-DS-I Dataset; (b) ASL-DS-II Dataset; (c) ASL-DS-III Dataset.

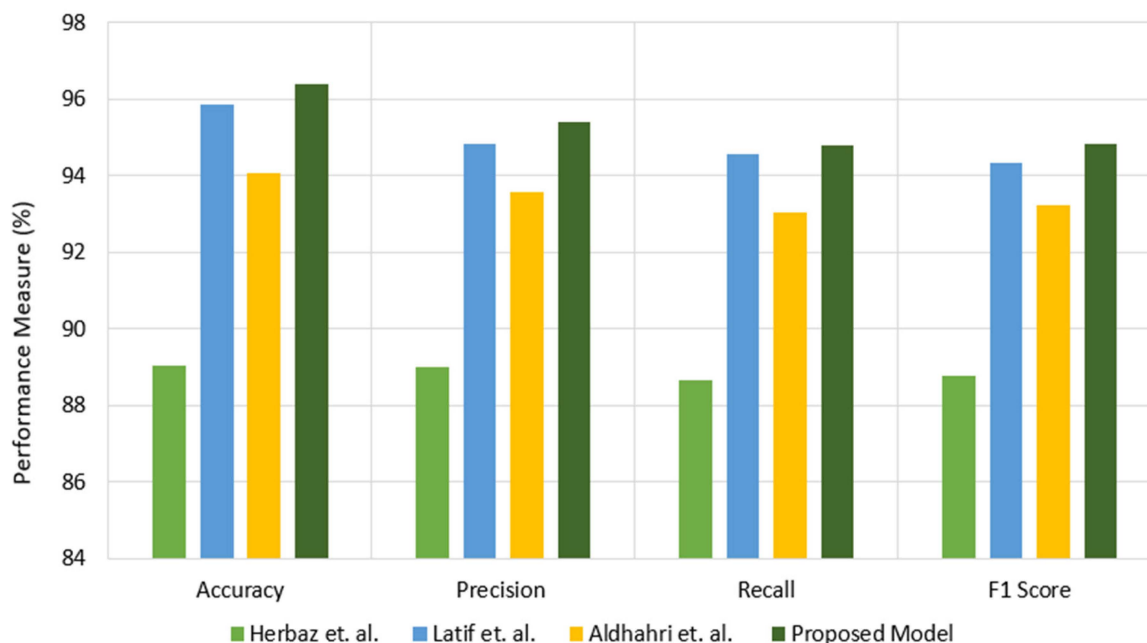


Figure 11. Comparatives analysis of proposed model with baseline methods (baseline methods needed to be changed) [13,29,30].

5. Conclusions

Sign language is essential for the deaf community, primarily using hand gestures for communication. However, a significant barrier exists for those outside this community, as they often find it challenging to understand sign language without an interpreter. This gap underscores the importance of technological solutions. Advancements in computer vision, especially in motion and gesture recognition, have been propelled forward by deep learning techniques. This study leverages these advancements to address communication challenges. We utilized the ArASL2018, which comprises images representing each sign of the Arabic alphabet, to train our model. Experimental analysis of the model yielded a promising recognition accuracy of 94.46%. For future work, we aim to expand the model's capabilities to include more complex sign language constructs beyond individual letters, such as words and phrases. Additionally, improving the model's performance in real-world scenarios, where background noise and varying lighting conditions can affect recognition accuracy, will be a focus. Another significant area of development is enhancing the model's adaptability to different sign language dialects, recognizing the diversity within the deaf community. Lastly, integrating this technology into accessible applications and devices to facilitate real-time translation and communication for the deaf and hearing-impaired individuals is a key goal.

Author Contributions: S.A.A.: conceptualization, methodology, writing—original draft, F.M.: software, writing—review and editing, H.A.D.: visualization, supervision. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the King Salman Center for Disability Research for funding this work through Research Group no KSRG-2023-512.

Data Availability Statement: The data presented in this study are openly available.

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

References

1. Agrawal, S.; Jalal, A.; Tripathi, R. A survey on manual and non-manual sign language recognition for isolated and continuous sign. *Int. J. Appl. Pattern Recognit.* **2016**, *3*, 99–134. [[CrossRef](#)]
2. Demircioglu, B.; Bulbul, G.; Kose, H. Turkish Sign Language recognition with Leap Motion. In Proceedings of the 24th Signal Processing and Communication Application Conference (SIU), Zonguldak, Turkey, 16–19 May 2016; pp. 589–592.
3. Hdioud, B.; Tirari, M.E.H. Facial expression recognition of masked faces using deep learning. *IAES Int. J. Artif. Intell.* **2023**, *12*, 921–930. [[CrossRef](#)]
4. Das, S.P.; Talukdar, A.K.; Sarma, K.K. Sign Language Recognition Using Facial Expression. *Procedia Comput. Sci.* **2015**, *58*, 210–216. [[CrossRef](#)]
5. Waters, D.; Campbell, R.; Capek, C.M.; Woll, B.; David, A.S.; McGuire, P.K.; Brammer, M.J.; MacSweeney, M. Fingerspelling, signed language, text and picture processing in deaf native signers: The role of the mid-fusiform gyrus. *Neuroimage* **2007**, *35*, 1287–1302. [[CrossRef](#)] [[PubMed](#)]
6. da Silva, E.P.; Kumada, K.M.O.; Costa, P.D.P. Analysis of Facial Expressions in Brazilian Sign Language (Libras). *Eur. Sci. J. ESJ* **2021**, *17*, 1. [[CrossRef](#)]
7. Kumar, K. DEAF-BSL: Deep IEARning Framework for British Sign Language recognition. *Trans. Asian Low-Resour. Lang. Inf. Process.* **2022**, *21*, 1–14. [[CrossRef](#)]
8. Jiang, X.; Satapathy, S.C.; Yang, L.; Wang, S.H.; Zhang, Y.D. A Survey on Artificial Intelligence in Chinese Sign Language Recognition. *Arab. J. Sci. Eng.* **2020**, *45*, 9859–9894. [[CrossRef](#)]
9. Saleh, Y.; Issa, G.F. Arabic sign language recognition through deep neural networks fine-tuning. *Int. J. Online Biomed. Eng.* **2020**, *16*, 71–83. [[CrossRef](#)]
10. Hasasneh, A. Arabic sign language characters recognition based on a deep learning approach and a simple linear classifier. *Jordanian J. Comput. Inf. Technol.* **2020**, *6*, 281–290. [[CrossRef](#)]
11. Hdioud, B.; Tirari, M.E.H. A Deep Learning based Approach for Recognition of Arabic Sign Language Letters. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 424–430. [[CrossRef](#)]
12. Bani Baker, Q.; Alqudah, N.; Alsmadi, T.; Awawdeh, R. Image-Based Arabic Sign Language Recognition System Using Transfer Deep Learning Models. *Appl. Comput. Intell. Soft Comput.* **2023**, *2023*, 5195007. [[CrossRef](#)]
13. Herbaz, N.; El Idrissi, H.; Badri, A. A Novel Approach for Recognition and Classification of Hand Gesture Using Deep Convolution Neural Networks. In Proceedings of the International Conference on Intelligent Systems and Pattern Recognition, New York, NY, USA, 11–13 May 2023; Springer Nature: Cham, Switzerland, 2023; pp. 90–105.
14. Altememe, M.S.; Abbadi, N.K.E. A hybrid model between a one-dimensional convolution neural network and machine learning algorithms for arabic sign language word recognition. In *AIP Conference Proceedings, Proceedings of the 4th International Scientific Conference of Alkafeel University (ISCKU 2022), Najaf, Iraq, 20–21 December 2022*; AIP Publishing: College Park, MD, USA, 2023; Volume 2977.
15. Alharthi, N.M.; Alzahrani, S.M. Vision Transformers and Transfer Learning Approaches for Arabic Sign Language Recognition. *Appl. Sci.* **2023**, *13*, 11625. [[CrossRef](#)]
16. Nahar, K.M.; Almomani, A.; Shatnawi, N.; Alauthman, M. A Robust Model for Translating Arabic Sign Language into Spoken Arabic Using Deep Learning. *Intell. Autom. Soft Comput.* **2023**, *37*, 2037–2057. [[CrossRef](#)]
17. Dabwan, B.A.; Jadhav, M.E.; Ali, Y.A.; Olayah, F.A. Arabic Sign Language Recognition Using EfficientnetB1 and Transfer Learning Technique. In Proceedings of the IEEE 2023 International Conference on IT Innovation and Knowledge Discovery (ITIKD), Manama, Bahrain, 8–9 March 2023; pp. 1–5.
18. Mohammed, H.I.; Waleed, J. Hand gesture recognition using a convolutional neural network for arabic sign language. In *AIP Conference Proceedings*; AIP Publishing: College Park, MD, USA, 2023; Volume 2475.
19. Benkaddour, M.K.; Abazi, Y.; Aziza Akram, Z. Hand Gesture and Sign Language Recognition Based on Deep Learning. Doctoral Dissertation, University of KASDI Merbah Ouargla, Ouargla, Algeria, 2023.
20. Al-onazi, B.B.; Nour, M.K.; Alshahran, H.; Elfaki, M.A.; Alnfai, M.M.; Marzouk, R.; Othman, M.; Sharif, M.M.; Motwakel, A. Arabic Sign Language Gesture Classification Using Deer Hunting Optimization with Machine Learning Model. *Comput. Mater. Contin.* **2023**, *75*, 3413–3429. [[CrossRef](#)]
21. Latif, G.; Mohammad, N.; Alghazo, J.; AlKhalaf, R.; AlKhalaf, R. ArASL: Arabic alphabets sign language dataset. *Data Brief* **2019**, *23*, 103777. [[CrossRef](#)]
22. Gal, M.S.; Rubinfeld, D.L. Data standardization. *NYUL Rev.* **2019**, *94*, 737. [[CrossRef](#)]
23. Benkaddour, M.K. CNN based features extraction for age estimation and gender classification. *Informatika* **2021**, *45*, 697–703. [[CrossRef](#)]
24. Trosten, D.J.; Sharma, P. Unsupervised feature extraction—A cnn-based approach. In Proceedings of the Image Analysis: 21st Scandinavian Conference, SCIA 2019, Norrköping, Sweden, 11–13 June 2019; Proceedings 21; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 197–208.
25. Zhao, X.; Wei, H.; Wang, H.; Zhu, T.; Zhang, K. 3D-CNN-based feature extraction of ground-based cloud images for direct normal irradiance prediction. *Sol. Energy* **2019**, *181*, 510–518. [[CrossRef](#)]
26. Chen, R.; Pan, L.; Zhou, Y.; Lei, Q. Image retrieval based on deep feature extraction and reduction with improved CNN and PCA. *J. Inf. Hiding Priv. Prot.* **2020**, *2*, 67. [[CrossRef](#)]

27. Yuan, Z.W.; Zhang, J. Feature extraction and image retrieval based on AlexNet. In Proceedings of the Eighth International Conference on Digital Image Processing (ICDIP 2016), Chengdu, China, 20–22 May 2016; Volume 10033, pp. 65–69.
28. Zhang, L.; Li, H.; Zhu, R.; Du, P. An infrared and visible image fusion algorithm based on ResNet-152. *Multimed. Tools Appl.* **2022**, *81*, 9277–9287. [[CrossRef](#)]
29. Latif, G.; Mohammad, N.; AlKhalaf, R.; AlKhalaf, R.; Alghazo, J.; Khan, M. An automatic Arabic sign language recognition system based on deep CNN: An assistive system for the deaf and hard of hearing. *Int. J. Comput. Digit. Syst.* **2020**, *9*, 715–724. [[CrossRef](#)]
30. Aldahri, E.; Aljuhani, R.; Alfaidi, A.; Alshehri, B.; Alwadei, H.; Aljojo, N.; Alshutayri, A.; Almazroi, A. Arabic sign language recognition using convolutional neural network and mobilenet. *Arab. J. Sci. Eng.* **2023**, *48*, 2147–2154. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.