sensors

MDPI

Article

Real-Time Hand Gesture Monitoring Model Based on MediaPipe's Registerable System

Yuting Meng [1,*], Haibo Jiang [2], Nengquan Duan [1] and Haijun Wen [1,*]

[1] College of Mechanical Engineering, North University of China, Taiyuan 030051, China; duannq@nuc.edu.cn
[2] SIMITECH Co., Xi'an 710086, China; mr.boj@outlook.com
* Correspondence: nataliaboolean@163.com (Y.M.); whjnuc@126.com (H.W.)

Abstract: Hand gesture recognition plays a significant role in human-to-human and human-to-machine interactions. Currently, most hand gesture detection methods rely on fixed hand gesture recognition. However, with the diversity and variability of hand gestures in daily life, this paper proposes a registerable hand gesture recognition approach based on Triple Loss. By learning the differences between different hand gestures, it can cluster them and identify newly added gestures. This paper constructs a registerable gesture dataset (RGDS) for training registerable hand gesture recognition models. Additionally, it proposes a normalization method for transforming hand gesture data and a FingerComb block for combining and extracting hand gesture data to enhance features and accelerate model convergence. It also improves ResNet and introduces FingerNet for registerable single-hand gesture recognition. The proposed model performs well on the RGDS dataset. The system is registerable, allowing users to flexibly register their own hand gestures for personalized gesture recognition.

Keywords: machine learning; real-time monitoring; gesture recognition; Triple Loss

Citation: Meng, Y.; Jiang, H.; Duan, N.; Wen, H. Real-Time Hand Gesture Monitoring Model Based on MediaPipe's Registerable System. *Sensors* **2024**, *24*, 6262. https://doi.org/10.3390/s24196262

Academic Editor: Stefanos Kollias

Received: 13 August 2024
Revised: 24 September 2024
Accepted: 25 September 2024
Published: 27 September 2024

1. Introduction

In daily life, gestures provide an efficient means of communication and play a significant role in interpersonal interactions [1]. With the advancement of technology and the development of human–computer interactions, gesture recognition has gradually gained attention as a natural and intuitive way of interacting [2]. Gesture recognition refers to the analysis and identification of human hand movements and postures by computers, enabling natural interactions between humans and machines. It is a vital research area in the fields of human–computer interaction and computer vision [3], with a wide range of potential applications, such as virtual reality, smart homes, security monitoring, handwriting recognition, medical image analysis, and more. Gesture recognition technologies can be classified into two main categories: sensor-based gesture recognition and image-based gesture recognition [4].

In the early stages, gesture recognition heavily relies on sensor-based methods due to immature hardware and algorithms. Data gloves or electromagnetic waves are commonly used to capture hand movements. For instance, IBM introduced a device called "DataGlove" [5], which monitored hand movements in real-time and transmitted them to computers. While this approach often provided more accurate results, the need to wear sensor devices limited its widespread adoption in everyday applications.

In recent years, with the rapid development of computer vision technology, the use of visuals in the industry is becoming increasingly widespread [6–8]. The methods for recognizing target movements and postures through image or video analysis are becoming more mature. Compared to traditional methods, image-based gesture recognition does not require additional wearable devices, making the recognition process more natural and gradually becoming mainstream technology [9]. In 2002, a research team at the

Massachusetts Institute of Technology (MIT) proposed a vision-based gesture recognition system that used infrared cameras to capture hand movements and employed computer vision techniques for gesture recognition [10]. Subsequently, products such as the Microsoft Kinect sensor and Leap Motion gesture recognizer have made gesture recognition more accessible to the general public [11]. This approach does not require extra equipment and can easily integrate with computers and mobile devices, making it increasingly popular.

Gesture recognition techniques include hand pose representation, feature extraction, and classification algorithms [12]. Hand pose representation methods are typically based on key points, skeletal data, or depth images. Feature extraction methods can utilize shape, texture, or motion-related features of gestures. Classification algorithms may involve traditional machine learning techniques such as SVM (Support Vector Machine) or Random Forests, as well as deep learning algorithms like CNN (Convolutional Neural Networks) or RNN (Recurrent Neural Network) [13].

Image-based gesture recognition can be further classified into two categories: static gesture recognition and dynamic gesture recognition [14,15]. Dynamic gestures refer to gesture sequences over time and space, where hand shape and position change with time. It emphasizes the process of gesture changes. On the other hand, static gestures refer to the posture of the hand at a specific moment, mainly including hand shape, texture, orientation, and relative spatial positions. As static gestures represent a certain state in the process of dynamic gestures, they form the core of gesture recognition. Thus, this paper focuses on exploring static gesture recognition.

Existing methods for static gesture recognition mainly involve custom modeling through traditional approaches and feature extraction and recognition through deep learning methods [16]. Currently, available deep learning models have enhanced the performance of gesture recognition to a certain extent. However, adding a new gesture requires retraining the existing models, and each new gesture demands a substantial amount of data for support. Current methods treat each gesture as a separate class, similar to distinguishing between apples and cats. However, in reality, each gesture falls under the broader category of "hand", and they share many common characteristics. Therefore, we propose using neural networks to train a classifier for the "hand" category, learning the differences between various gestures, and when encountering a new gesture, the network can recognize and classify it accordingly. By inputting a diverse set of gestures into the neural network, we enabled the model to discern the commonalities and essence of different gestures and learn the distinctions between various and similar gestures. Ultimately, this approach achieves the recognition of new gestures, completing a register-based gesture recognition system.

2. Related Work

There are many existing datasets for sign language recognition tasks. In this section, we first review some existing sign language datasets. Athitsos et al. [17] proposed an American sign language dataset called ASLLVD, consisting of 9800 video samples containing 3300 sign language words. Sincan and Keles proposed a Türkiye sign language dataset called AUTSL, which contained 38,336 video samples, including 226 sign languages performed by 43 different sign language speakers. These videos were recorded in both indoor and outdoor environments. This dataset has color, depth, and skeleton modes. Necati [18] proposed RWTH-PHOENIX-WEATHER-2014T, a German sign language dataset for continuous sign language recognition, which was based on weather forecast videos broadcasted by nine sign language hosts. The training set, validation set, and test set of the dataset contain 7096, 519, and 642 data samples, respectively. The CSL-Daily [19] dataset 3 can be used for continuous sign language recognition and translation tasks. CSL-Daily focuses more on daily life scenarios, including multiple themes such as family life, healthcare, and school life. The training, validation, and testing sets of CSL-Daily contain 18,401, 1077, and 1176 video samples, respectively. These datasets have their own focuses, but they still vary from the different posture situations of the same gesture that this article wants to solve.

Therefore, in subsequent work, this article has created a small gesture dataset RGDS to supplement the existing gesture dataset with fewer training samples for different postures of the same gesture.

Generally speaking, constructing gesture recognition and classification models is the main stage of gesture recognition technology. By extracting the spatiotemporal features of gestures and classifying them, the existing technologies can mainly be divided into two categories: traditional methods and deep learning. Most traditional methods use pre-defined templates or sequence similarity matchingschemes. Among them, traditional methods such as He Li et al. [14] used the maximum likelihood criterion Hausdorff distance for recognition [15], and used a multi-resolution search strategy to improve the search speed while also recognizing letter gestures as well. However, the recognition effect is not good for gestures that undergo deformation (rotation and scaling). Zhang Liangguo et al. [20] sed the Hausdorff distance template matching method to compare the contour feature points of gesture areas and successfully completed the recognition of 30 Chinese sign language gestures; Yang Xuewen et al. [21] overcame the shortcomings of existing gesture recognition algorithms in handling gestures such as scaling, translation, and rotation, using a method that combines gesture main direction and class Hausdorff distance. Zhu Jiyu et al. [22] used a gesture recognition method based on structural analysis to obtain the overall and local change information of gestures, increasing the types of recognizable gesture types. However, these traditional methods also have common drawbacks, such as relatively complex calculation processes, high manpower consumption, and unsatisfactory real-time performance.

Deep learning methods have now become the mainstream method for completing gesture recognition work: experimental results from studies [23–29] have shown that gesture recognition technology based on neural network training has the ability to improve the accuracy of some meaningful gesture recognition work to over 95%. Xin Wenbin et al. [23] used a static gesture real-time recognition method based on the ShuffleNetv2-YOLOv3 model, extracted features from gesture images using ShuffleNetv2, and then classified gestures using the YOLOv3 neural network model, increasing the fps from 41 to 44. By adopting the CBAM attention mechanism module, the model's accuracy increased from 96.6% to 98.2%. Wu [25] adopted a novel recognition algorithm of a dual channel convolutional neural network (DC-CNN) to simultaneously extract features from gesture images and hand images, effectively improving the generalization ability of CNN, but the improvement in accuracy was not very significant. At this point, people noticed the shortcomings of single visual recognition in reading gestures. Songyao Jiang et al. [27] proposed independent sign language recognition (SAM-SLR-v2), a skeleton-aware multimodal framework with a global synthesis model. This pattern provides annotations for skeleton-based sign language recognition, but the recognition performance was poor in cases of occlusion or insignificant posture. Papadimitriou et al. [28] proposed a deep learning framework 3D-DCNN + ST-MGCN using appearance and skeletal information for automatic sign language recognition without special input, which reduced the relative error rate of Greek language recognition by 53%. However, this was only for the recognition of the entire arm and did not show any improvement compared to single-finger or hand recognition.

It can be found in the literature that current deep learning gesture recognition work has developed quite maturely. However, current image-based gesture recognition is based on recognizing the projected image of the hand on the image screen. However, what we generally consider gestures is based on the mutual combination of finger joints, i.e., the same gesture forms different images on the image screen in different poses. Therefore, image-based gesture recognition has limitations in recognizing the same gesture in different poses. In response to the above issues, this method obtains the three-dimensional coordinates of the finger's joint points and identifies the features of the joint points, enabling users to recognize the same gesture in different postures. Considering the classification of features in gesture recognition models, we did not adopt the commonly used softmax classifier. We found that Florian Schroff et al. [30] used Triple Loss to construct FaceNet, which directly

learned the mapping from facial images to a compact Euclidean space, where distance directly corresponded to the measure of facial similarity. The goal of Triple Loss is to make the embeddings of samples with the same label as close as possible in the embedding space and the embeddings of samples with different labels as far apart as possible. The Softmax loss function is commonly used for multi-class classification tasks. It maps the output of the network to a probability distribution such that the sum of probabilities for each category is one. The objective of the Softmax loss function is to maximize the probability of the correct category while minimizing the probability of the incorrect category. However, the Softmax loss function may be troubled when dealing with large intra-class differences as it only considers the differences between samples of the same category and ignores the differences between different categories. In contrast, the Softmax loss function mainly focuses on the differences between samples of the same category, while the Triple Lossfunction pays more attention to increasing the similarity between samples of the same category and increasing the degree of difference between different categories.

Finally, the MediaPipe we used in our research is a cross-platform machine learning framework open-source by Google [31] designed to assist developers in building machine learning applications based on visual, audio, and sensor inputs. Finger in MediaPipe is an algorithm module used for hand pose estimation [32]. It can locate and track fingers through input hand images and estimate the three-dimensional posture of fingers. The input of the Finger module is a set of hand images, such as hand images captured by a camera or hand images that have already been captured. The Finger module detects and tracks fingers, obtaining the position and posture information of each finger. Based on deep neural networks, it trains on a large amount of hand image data to achieve the accurate detection and tracking of fingers. In addition, the Finger module also utilizes some image processing and computer vision algorithms, such as morphological processing, filters, and Kalman filters, to further improve the accuracy and stability of detection and tracking.

3. Research Methodology

3.1. Gesture Recognition Process

In the context of image recognition and classification, there are four main tasks: classification, localization, detection, and segmentation. For gesture recognition, we needed to address the tasks of classification, localization, and detection. To achieve this, we used the following steps:

1. Classification and Localization:

Utilize the open-source MediaPipe provided by Google to obtain the position and coordinate points of the hand. This step helps us identify and locate the hand within the image.

2. Data Preprocessing:

Apply geometric transformations to the coordinate points of the hand obtained in the previous step to normalize the data. This normalization step ensures that the hand's shape and position are consistent and comparable across different images.

3. Feature Embedding:

Use FingerNet to perform embedding on normalized hand data. FingerNet is a deep neural network designed to extract representative features from the hand's coordinate points.

4. Loss Functions:

Apply the Triple Loss and CrossEntropy Loss functions during the network training phase. The Triple Loss function is responsible for pushing embeddings of samples with the same gesture label closer together in the embedding space and pushing embeddings of samples with different gesture labels farther apart. The CrossEntropy Loss function is utilized for the classification aspect of network training.

By following these steps (Figure 1), the gesture recognition model can be trained based on the features extracted from hand coordinates, leading to accurate classifying. This approach combines localization and classification to achieve effective gesture recognition.

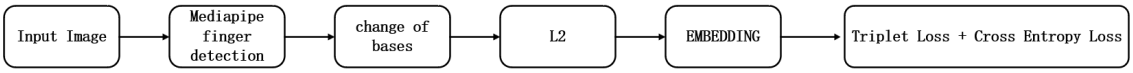

Figure 1. Gesture classification implementation process.

3.2. Data Preparation

For this study, gesture recognition was based on the key points of the fingers. Therefore, if the bending degree of finger joints is the same, it can be considered the same gesture. To address this feature, a registerable gesture dataset (RGDS) was constructed specifically for gesture recognition. The RGDS consists of hand gesture images with a resolution of 2624 × 2457. It contains 32 different gesture categories, each with 50 samples, resulting in a total of 1600 samples.

As shown in Figure 2, (1) and (2) represent different postures of the same gesture, while (1) and (2) depict different gestures altogether. The RGDS dataset captures these variations in gesture postures and includes images with various finger joint bending degrees, enabling the model to recognize the same gesture in different poses.

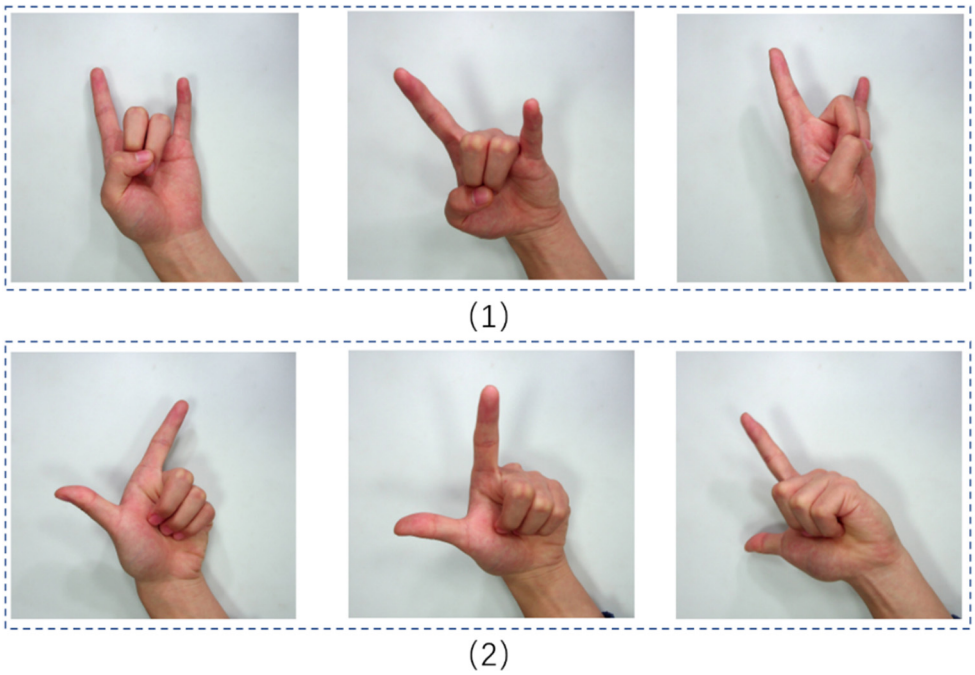

Figure 2. Gesture data. (1) and (2) represent gesture photographs for two different gestures.

Please note that the RGDS dataset has been curated to encompass diverse hand gesture samples, facilitating effective training and the evaluation of the gesture recognition model.

3.3. Data Preprocessing

After processing the images containing hands through the MediaPipe network, we obtained the three-dimensional coordinates of 21 key points of the fingers, as shown in Figure 3. At this point, the x and y coordinates of the three-dimensional points are based on the image's left-bottom corner as the coordinate origin. The z-coordinate is based on the 0th point as the coordinate origin.

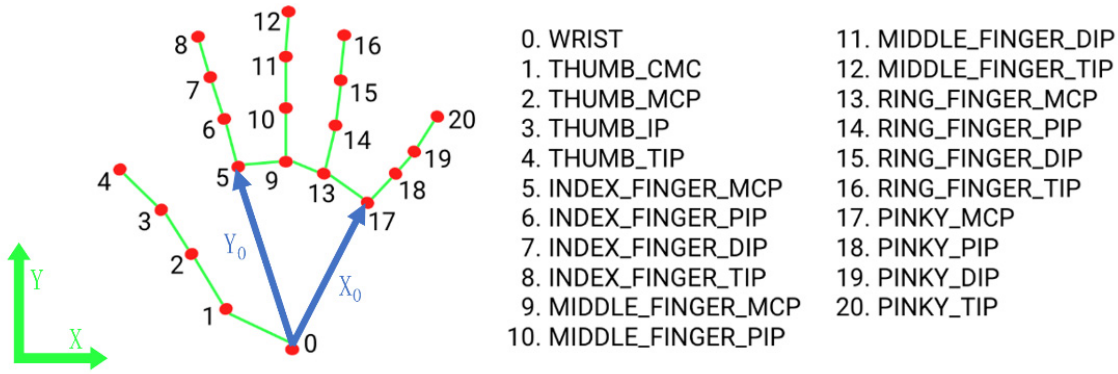

Figure 3. MediaPipe finger landmark. The red dots are the 21 key points selected for the hand, which are connected by a green line to form a complete line of identification of the hand.

To facilitate further image processing, we standardized and rectified the coordinates of the 21 key points of the fingers. Here, we take the 0th point as the coordinate origin; the x-axis runs from 0 to 17, and the y-axis runs from 0 to 5. Using the right-hand rule, we can obtain a three-dimensional coordinate system based on the 0th point, which will be used for subsequent processing.

Canonical transformations are mathematical transformations commonly used to analyze and simplify the description of physical systems. They involve selecting appropriate coordinate system transformations to convert physical quantities in the original coordinate system, making the problem's formulation more concise or convenient for solving. The core idea of canonical transformations is to choose a new set of basis vectors to represent vectors in the original coordinate system. These new basis vectors often possess special properties, such as orthogonality or normalization, to simplify the problem's description or solution.

In this specific case, we are considering two sets of basis vectors: one based on the points 0 to 5 and another based on the points 0 to 17.

(1) Taking the 0th point as the coordinate origin, we performed a translation of the original coordinates. Let x_0, y_0 represent the original coordinates of point 0, and x_i, y_i represent the original coordinates of point i. After the translation, we obtained new coordinates X_0, Y_0.

$$X_0 = x_i - x_0 \tag{1}$$

$$Y_0 = y_i - y_0 \tag{2}$$

(2) We calculated the vector $\vec{P_0P_{17}}$ and $\vec{P_0P_5}$

$$\vec{P_0P_{17}} = ((x_{17} - x_0), (y_{17} - y_0), (z_{17} - z_0)) \tag{3}$$

$$\vec{P_0P_5} = ((x_5 - x_0), (y_5 - y_0), (z_{17} - z_0)) \tag{4}$$

(3) We calculated the normal vector $\vec{P_z}$ of the plane formed by vectors $\vec{P_0P_5}$ and $\vec{P_0P_{17}}$

$$\vec{P_z} = \vec{P_0P_5} \times \vec{P_0P_{17}} \tag{5}$$

$$\vec{P_z} = \begin{vmatrix} i & j & k \\ x_5 - x_0 & y_5 - y_0 & z_5 - z_0 \\ x_{17} - x_0 & y_{17} - y_0 & z_{17} - z_0 \end{vmatrix} \tag{6}$$

(4) Change in Basis

For a point P in the basis of $\vec{Q} = \left(\vec{a}, \vec{b}, \vec{c}\right)$, its coordinates are given by (x_r, y_r, z_r) using Equation (7). In the basis of $\vec{Q} = \left(\vec{A}, \vec{B}, \vec{C}\right)$, its coordinates are given by (x_q, y_q, z_q) using Equation (8). The coordinates of $\vec{A}$, $\vec{B}$, and $\vec{C}$ in the basis of $\vec{R}$ are given, respectively, by (X_{ar}, Y_{ar}, Z_{ar}), (X_{br}, Y_{br}, Z_{br}), and (X_{cr}, Y_{cr}, Z_{cr}) using Equations (9)–(11). Formula (12) can be obtained by Formulas (9)–(11); we obtained Equation (13), which represents the coordinates of point P in both bases, showing that (7) = (8). From this, we could derive the coordinate transformation matrix F (14) that transforms point P from $\vec{R}$ to $\vec{Q}$.

$$P = \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} \cdot \vec{R} = \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} \cdot \left[\vec{a}, \vec{b}, \vec{c}\right] = x_r \vec{a} + y_r \vec{b} + z_r \vec{c} \tag{7}$$

$$P = \begin{bmatrix} x_q \\ y_q \\ z_q \end{bmatrix} \cdot \vec{Q} = \begin{bmatrix} x_q \\ y_q \\ z_q \end{bmatrix} \cdot \left[\vec{A}, \vec{B}, \vec{C}\right] = x_q \vec{A} + y_q \vec{B} + z_q \vec{C} \tag{8}$$

$$\vec{A} = \begin{bmatrix} X_{ar} \\ Y_{ar} \\ Z_{ar} \end{bmatrix} \cdot \vec{P} = X_{ar} \vec{a} + Y_{ar} \vec{c} + Z_{ar} \vec{c} \tag{9}$$

$$\vec{B} = \begin{bmatrix} X_{br} \\ Y_{br} \\ Z_{br} \end{bmatrix} \cdot \vec{P} = X_{br} \vec{a} + Y_{br} \vec{c} + Z_{br} \vec{c} \tag{10}$$

$$\vec{C} = \begin{bmatrix} X_{cr} \\ Y_{cr} \\ Z_{cr} \end{bmatrix} \cdot \vec{P} = X_{cr} \vec{a} + Y_{cr} \vec{c} + Z_{cr} \vec{c} \tag{11}$$

$$\vec{Q} = \left[\vec{A}, \vec{B}, \vec{C}\right] = \begin{bmatrix} X_{ar} & X_{br} & X_{cr} \\ Y_{ar} & Y_{br} & Y_{cr} \\ Z_{ar} & Z_{br} & Z_{cr} \end{bmatrix} \cdot \vec{P} \tag{12}$$

$$\begin{bmatrix} x_q \\ y_q \\ z_q \end{bmatrix} = \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} \cdot \vec{P} \cdot \vec{Q}^{-1} = \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} \cdot F \tag{13}$$

$$F = \vec{P} \cdot \vec{Q}^{-1} \tag{14}$$

3.4. Model Construction

After data preprocessing, gesture images were transformed into $1 \times 22 \times 3$-dimensional data. The 22 dimensions comprise the following information: the first dimension represents the hand's orientation, dimensions 2 to 6 contain the coordinates of each joint of the thumb, dimensions 7 to 10 contain the coordinates of each joint of the index finger, dimensions 11 to 14 contain the coordinates of each joint of the middle finger, dimensions 15 to 18 contain the coordinates of each joint of the ring finger, and dimensions 19 to 22 contain the coordinates of each joint of the little finger. These gesture features are then extracted and combined using the FingerComb block to form $1 \times 33 \times 32$dimensional data. The structure of the FingerComb block is illustrated in Figure 4, where convolution is performed using conv1d.

The main innovation of ResNet-16 lies in the introduction of residual connections, which addresses the issues of gradient vanishing and model degradation during the training of deep networks. This shortcut connection directly adds the input features to the output features, enabling information to flow directly through the network facilitating easier training and optimization.

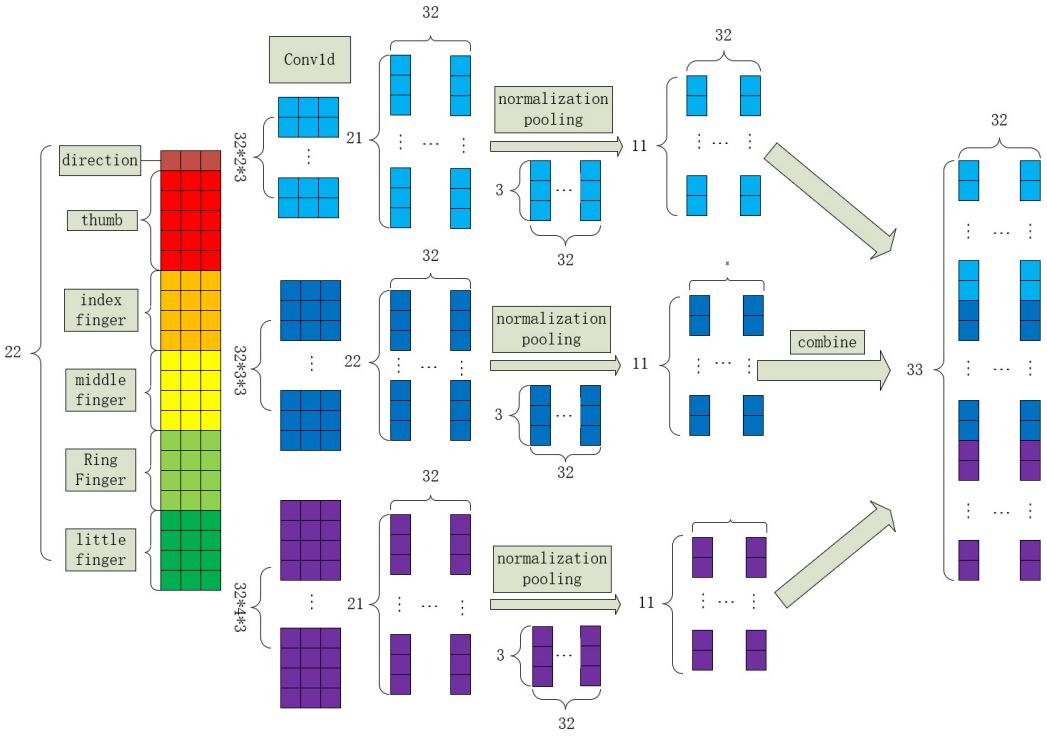

Figure 4. FingerComb block.

In this study, an improved version of ResNet-16, called FingerNet, was achieved by incorporating the FingerComb Block, as shown in Figure 5. The model's parameters are listed in Table 1. FingerNet takes $1 \times 22 \times 3$-dimensional data as the input and, through convolutional operations, yields a final 1×32-dimensional EMBEDDING.

Table 1. The architecture layer diagram of FingerNet.

Layer Name	Filter	Output Size
Input		$1 \times 22 \times 3$
FingerComb	$\begin{bmatrix} \mathbf{1 \times 2,} & \mathbf{32} & \mathbf{s = 1, p = 0} \\ \mathbf{1 \times 2,} & \mathbf{32} & \mathbf{s = 1, p = 0} \end{bmatrix}$ $\begin{bmatrix} \mathbf{1 \times 3,} & \mathbf{32} & \mathbf{s = 1, p = 1} \\ \mathbf{1 \times 3,} & \mathbf{32} & \mathbf{s = 1, p = 1} \end{bmatrix}$ $\begin{bmatrix} \mathbf{1 \times 4,} & \mathbf{32} & \mathbf{s = 1, p = 1} \\ \mathbf{1 \times 4,} & \mathbf{32} & \mathbf{s = 1, p = 1} \end{bmatrix}$	$1 \times 33 \times 32$
Conv2_x	$\begin{bmatrix} \mathbf{1 \times 3,} & \mathbf{32} & \mathbf{s = 1, p = 1} \\ \mathbf{1 \times 3,} & \mathbf{32} & \mathbf{s = 1, p = 1} \end{bmatrix} \times \mathbf{2}$	$1 \times 33 \times 32$
Conv3_x	$\begin{bmatrix} \mathbf{1 \times 3,} & \mathbf{64} & \mathbf{s = 2, p = 1} \\ \mathbf{1 \times 3,} & \mathbf{64} & \mathbf{s = 1, p = 1} \end{bmatrix}$ $\begin{bmatrix} \mathbf{1 \times 3,} & \mathbf{64} & \mathbf{s = 1, p = 1} \\ \mathbf{1 \times 3,} & \mathbf{64} & \mathbf{s = 1, p = 1} \end{bmatrix}$	$1 \times 17 \times 64$
Conv4_x	$\begin{bmatrix} \mathbf{1 \times 3,} & \mathbf{128} & \mathbf{s = 2, p = 1} \\ \mathbf{1 \times 3,} & \mathbf{128} & \mathbf{s = 1, p = 1} \end{bmatrix}$ $\begin{bmatrix} \mathbf{1 \times 3,} & \mathbf{128} & \mathbf{s = 1, p = 1} \\ \mathbf{1 \times 3,} & \mathbf{128} & \mathbf{s = 1, p = 1} \end{bmatrix}$	$1 \times 9 \times 128$
Conv5_x	$\begin{bmatrix} \mathbf{1 \times 3,} & \mathbf{256} & \mathbf{s = 2, p = 1} \\ \mathbf{1 \times 3,} & \mathbf{256} & \mathbf{s = 1, p = 1} \end{bmatrix}$ $\begin{bmatrix} \mathbf{1 \times 3,} & \mathbf{256} & \mathbf{s = 1, p = 1} \\ \mathbf{1 \times 3,} & \mathbf{256} & \mathbf{s = 1, p = 1} \end{bmatrix}$	$1 \times 5 \times 256$
Avgpool	256 K = 3, s = 1, p = 1	$1 \times 1 \times 256$
FC	(256, 128)	1×128
FC	(128, 32)	1×32

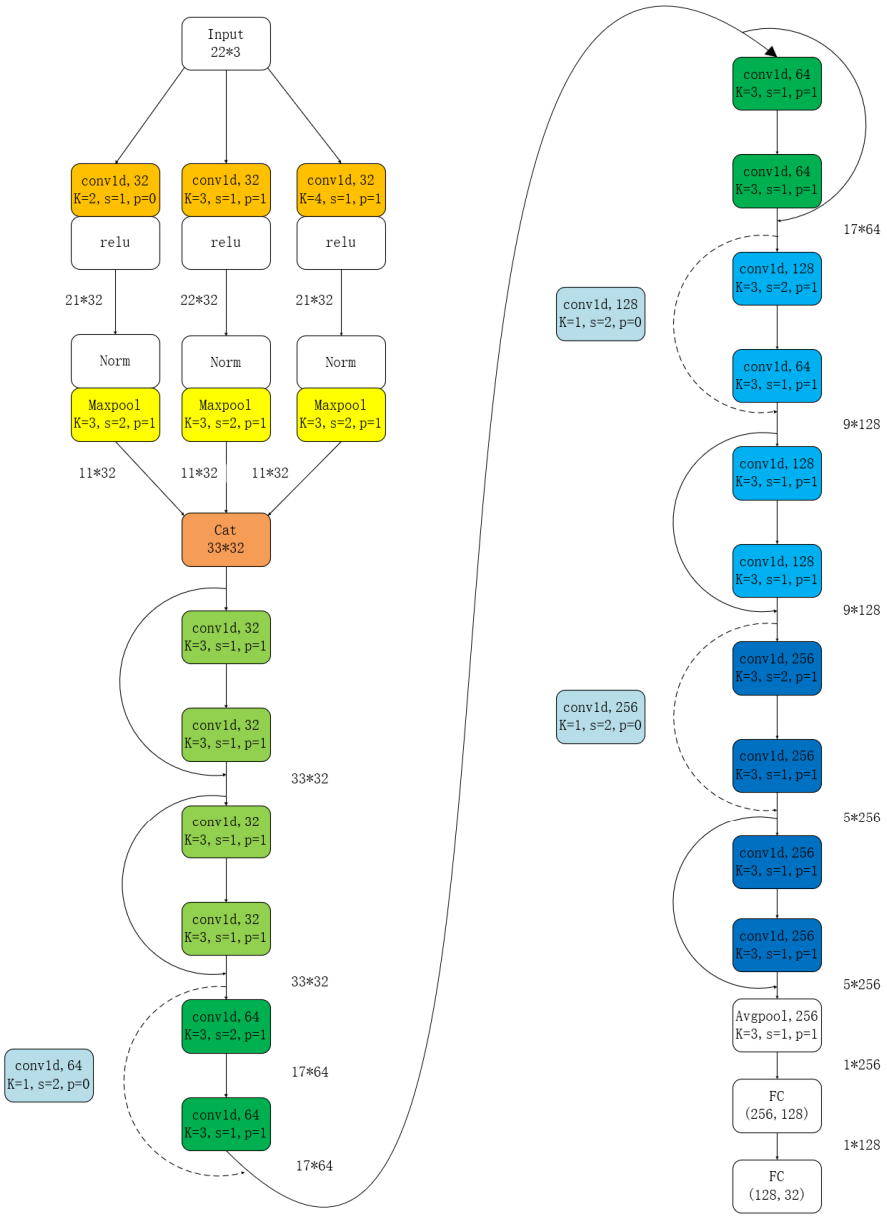

Figure 5. Structure of FingerNet.

3.5. Loss Function

The Triple Loss function enhances the discriminability of gesture features by introducing the concept of triplets. Each triplet consists of an anchor sample (a), a positive sample (p), and a negative sample (n). The objective of the Triple Loss is to minimize the distance between the anchor sample and the positive sample while maximizing the distance between the anchor sample and the negative sample, with the addition of a margin value to ensure increased dissimilarity. By optimizing the Triple Loss function, the network is compelled to learn gesture feature embeddings with better discriminability. Triple Loss performs well in addressing the issue of large intra-class variations, thus improving the accuracy of gesture recognition. However, in practical application, Triple Loss is difficult to train, and the model may struggle to converge. To overcome this, the training process simultaneously employs the CrossEntropy loss function with Triple Loss to improve the model's convergence speed and gesture recognition performance. This approach balances the optimization of intra-class similarity and inter-class dissimilarity.

Formula (15) represents the Triple Loss, where a denotes the anchor sample, p represents the positive sample, and n represents the negative sample. Formula (16) represents

CrossEntropy loss, where m is the number of samples, W_j represents the jth weight in the model parameters, y_i is the true label for each sample, and $\hat{y}_i$ is the model output. Formula (17) represents the total loss, which is the combination of Triple Loss and CrossEntropy Loss.

$$Lt = \max\{d(a,p) - d(a,n) + margin, 0\} \tag{15}$$

$$\mathrm{L(w)} = -\frac{1}{\mathrm{m}}\sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{y_i}(\hat{\mathrm{y}}_\mathrm{i}) + \frac{1}{\mathrm{m}}\sum_{\mathrm{i}=1}^{\mathrm{m}}(1-\mathrm{y_i})\log\left(\mathrm{e}^{\frac{1}{\mathrm{m}}\Sigma_{\mathrm{j}=1}^{\mathrm{m}}\mathrm{w_j}}\right) \tag{16}$$

$$L = Lt + L(\mathrm{w}) \tag{17}$$

4. Experimental Procedure and Results

In this section, we analyze the performance of the FingerNet model in the process of gesture recognition through experiments. This model utilizes the RGDS dataset, which we constructed ourselves. The dataset contained a total of 32 classes of gesture data, with 28 classes used as training data and the remaining four classes used for model validation. The hardware and software environment for model training in this study are as shown in Table 2. We built the FingerNet model using the PyTorch framework and trained it using RTX 8000 GPU.

Table 2. Hardware and software platforms.

Type	Type Specification
Operating system	Window 11
CPU	Xeon(R) Gold 5218
GPU	NVIDIA Quadro RTX 8000
Deep learning framework	Pytorch 1.13.0
Language	Python 3.7.15
Memory	256 GB
CUDA	11.7
cudnn	8500

After the completion of the model's construction, various training parameters needed to be set before training, as shown in Table 3. The model was trained for a total of 500 epochs, and the margin was a parameter for the Triple Loss. Additionally, the "min_tracking_confidence" is the minimum confidence threshold for hand tracking in MediaPipe finger detection, while the "min_detection_confidence" represents the minimum confidence threshold for hand detection. The "max_num_hands" parameter specifies the maximum number of hands to be recognized in the input data.

Table 3. Model training parameters.

Type	Parameter
epoch	500
BatchSize	256
margin	5
lr	0.2
min_tracking_confidence	0.8
min_detection_confidence	0.62
max_num_hands	1

After setting up the various training parameters, the model was trained, and Figure 6 shows the change in model training loss with respect to the training steps. Before 200k steps, the model's loss appears to be unstable but shows an overall convergence trend. After 200k steps, the model's loss mostly tends to approach zero; although there may be some fluctuations in the process, the loss values are very small.

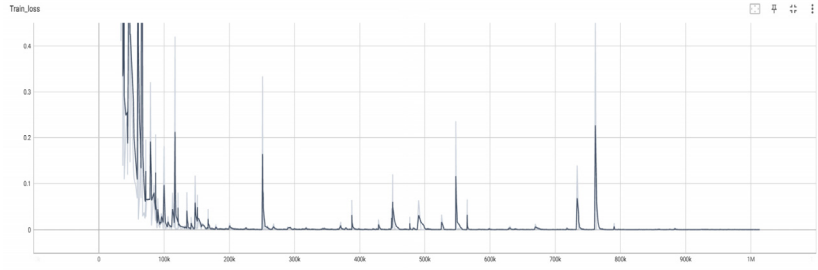

Figure 6. Training process.

After training the model, it was tested using the remaining four classes of hand gestures from the RGDS dataset as registered gestures. The model processed the data and performed cluster analysis on the gestures. The test results are shown in Figure 7, where the *x*-axis represents the 32 classes of hand gestures, and the *y*-axis represents the L2 distance. The figure contains four subplots, each representing the L2 distance between the currently registered gesture and each class in the test data.

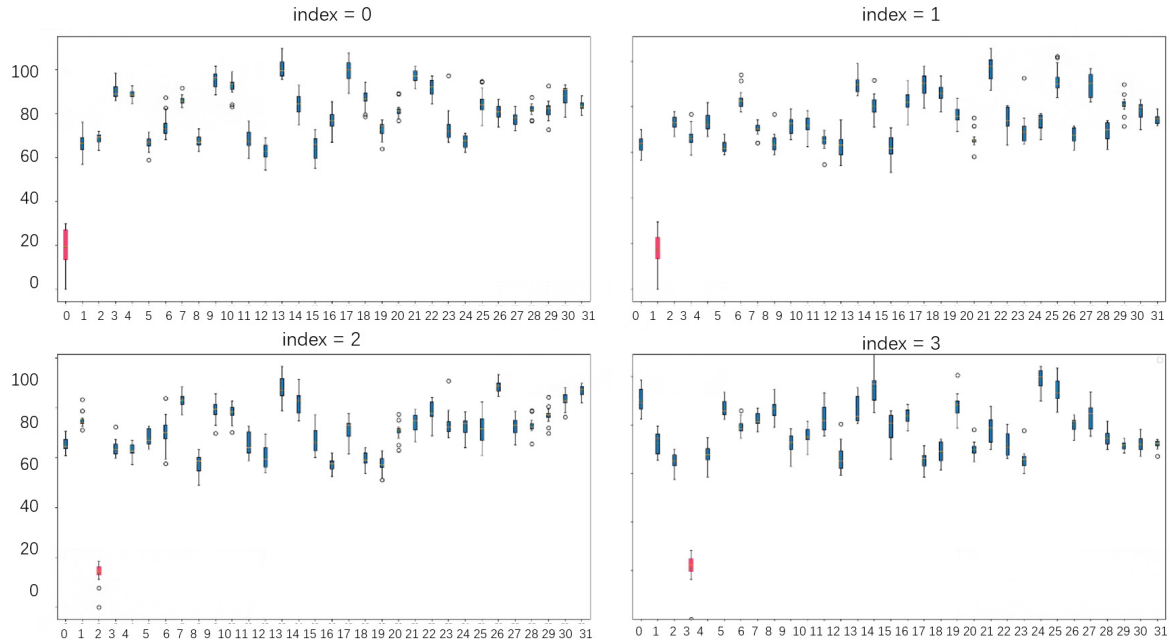

Figure 7. Test results box diagram. The parts marked in red are the gestures in this class that have the smallest L2 distance compared to the other gestures.

For example, in the first subplot, the registered gesture is labeled as the 0th class. From this plot, it can be observed that the L2 distance between the registered 0th class gesture and other 0th class gestures are the smallest, indicating that there is a high intra-class similarity. On the other hand, the distance between the registered gesture and gestures from other classes is larger, indicating a greater inter-class dissimilarity.

Additionally, each class of gestures is well-clustered, showing high intra-class cohesion. Therefore, when setting the threshold to 40, it is possible to completely distinguish between gestures of the same class as the registered gesture and gestures from other classes. This indicates that the FingerNet model is capable of effectively clustering and differentiating between hand gestures based on the learned gesture embeddings.

After training the model, real-time hand gesture processing and displays were performed using OpenCV to read images from the camera. The images were processed through MediaPipe and then passed to FingerNet for hand gesture recognition. The results are shown in Figure 8.

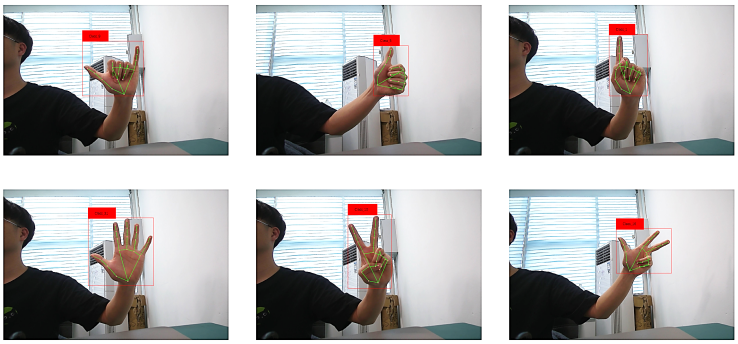

Figure 8. Real-time gesture detection.

From this figure, it can be observed that after processing by FingerNet, the model accurately detected the hand positions and hand gesture classes in real-time. Moreover, the model was able to recognize new hand gesture classes through the registration process.

The real-time processing and recognition capabilities of FingerNet demonstrate its effectiveness and practicality in hand gesture recognition applications. The model shows promising performance in accurately identifying hand gestures and can be extended to recognize a wide range of hand movements and poses.

In this paper, in order to further validate the effectiveness of the model method, we also compared some existing models with the model proposed in this chapter on the AUTSL dataset (Table 4), and the results are displayed in accuracy order in the table. Among them, the baseline RGB accuracy comes from the CNN-FPM BiLSTM attention model proposed by Sincan et al. [33], which extracts image features using CNN, fuses multi-scale diffusion convolution through FPM, collects time-domain features using BiLSTM with the attention mechanism, and finally performs time-domain pooling and gesture classifications. The accuracy of LSTM, Transformer, ST-GCN, and SL-GCN methods comes from the OpenHands framework proposed by Selvaraj et al. [34], which mainly uses pre-training and self-supervised learning to improve the accuracy of the model. The remaining models FE + LSTM, MViT-SLR, HWGAT, 3D-DCNN + ST-MGCN, SAM-SLR, and STF + LSTM are from the corresponding papers [28,35–40]. These models reflect the best results from the current AUTSL dataset. From the comparison results, it can be seen that the accuracy of the method proposed in this paper reaches 0.953. Although not the best results on this dataset, these models were able to effectively recognize large-scale gestures through multimodal skeleton data.

Table 4. Comparison results between our method and existing methods on AUTSL.

Method	Accuracy
Baseline RGB	0.425
Baseline RGB-D	0.632
LSTM	0.774
Transformer	0.810
ST-GCN	0.904
SL-GCN	0.919
FE + LSTM	0.934
MViT-SLR	0.957
HWGAT	0.958
3D-DCNN + ST-MGCN	0.984
SAM-SLR	0.985
STF + LSTM	0.986
FINGER-NET	0.953

The sample recording in the AUTSL dataset is clear, the lighting is uniform, the frame rate is high, and there is less dynamic blur, making it suitable for skeleton pose estimation. It can accurately obtain finger positions and recognize gestures using graph convolution methods. However, in some scenarios, the quality of the samples to be recognized is relatively low, which has a significant impact on the estimation of body and hand skeletons, limiting the accuracy of subsequent gesture recognition.

In addition, the video recording time in the ChaLearLAPISOGD gesture dataset is relatively early, the frame rate is relatively low, and there are many samples that lack lighting and blur, which makes skeleton estimation prone to containing many errors. We also conducted comparative tests on the model. Table 5 shows the accuracy comparison results of this method on the dataset.

Table 5. Comparison results between our method and existing methods on ChaLearn IsoGD.

Method	Accuracy
Baseline	0.241
Zhu et al. [41]	0.509
Wang et al. [42]	0.555
Li et al. [43]	0.569
FINGER-NET	0.572

At the same time, we also conducted further research on another contribution of this article, the registrable dataset RGDS. We tested the existing algorithm models and our model on our dataset and compared them in the smaller-scale dataset. The accuracy is shown in the following Table 6.

Table 6. Comparison results between our method and existing methods on RGDS.

Method	Accuracy
LSTM	0.574
Transformer	0.610
Wang et al. [42]	0.706
Li et al. [43]	0.745
StepNet [44]	0.821
FINGER-NET	0.878

5. Results and Analysis

In this paper, we constructed a registerable gesture recognition dataset (RGDS) containing 32 different gesture classes, each with 50 images, totaling 1600 gesture images. We proposed a normalization method based on canonical transformations for gesture data to facilitate feature extraction and combination. Additionally, we introduced the FingerComb block for feature extraction and combination, which improved the robustness of gesture features and accelerated model convergence. We made improvements to the ResNet architecture to create the FingerNet model and tested it on the RGDS dataset to evaluate the performance and feasibility of the proposed methods. A large number of experiments show that this method has high accuracy in registered gesture recognition, especially on the dataset we created ourselves with absolute accuracy. The advantage of this is that it provides an effective solution for gesture interaction and operation in practical application scenarios.

However, the current model in this article only focuses on the recognition of single gestures when they can be registered and has weak recognition ability for gestures involving multiple gestures or complex actions (such as gestures made with both hands simultaneously). Moreover, large-scale gesture recognition typically requires the inclusion of a wider range of gesture samples, and the generalization ability and accuracy of improvement of this model for large-scale gesture recognition are not very good. In the future, we plan to

further promote and improve this model by achieving the registrable recognition of hand gestures and enhancing the model's ability to understand complex gestures.

Author Contributions: The authors confirm their contributions to the paper as follows: Conceptualization and design of the study: Y.M., H.J., N.D. and H.W.; Data collection: H.J.; Analysis and interpretation of the results: N.D.; Drafting of the manuscript: Y.M. and H.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: For reasons (the study was not in the biomedical field, the hand data are not identifiable, and the source of the hand data was only the authors of this study), ethical review and approval was waived for this study.

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

Data Availability Statement: Data is unavailable due to privacy restrictions.

Conflicts of Interest: The authors declare that they have no conflict of interest to report regarding the present study. And Haibo Jiang is employed by SIMITECH company. The company played no role in the design of the study, the collection, analysis, or interpretation of data, the writing of the manuscript, or the decision to publish the article.

References

1. Jiang, D.; Zheng, Z.; Li, G.; Sun, Y.; Kong, J.; Jiang, G.; Xiong, H.; Tao, B.; Xu, S.; Yu, H.; et al. Gesture Recognition Based on Binocular Vision. *Cluster Comput.* **2019**, *22*, 13261–13271. [CrossRef]
2. Feng, Z.; Yang, B.; Zheng, Y.W.; Xu, T.; Tang, H.K. *Hand Tracking Based on Behavioral Analysis for Users*; China Science Publishing & Media Ltd.: Beijing, China, 2013; Volume 24, pp. 2101–2116.
3. Rautaray, S.S.; Agrawal, A. Vision based hand gesture recognition for human computer interaction: A survey. *Artif. Intell. Rev.* **2015**, *43*, 1–54. [CrossRef]
4. Kaaniche, M.B.; Bremond, F. Recognizing gestures by learning local motion signatures of HOG descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 2247–2258. [CrossRef] [PubMed]
5. Weissmann, J.; Salomon, R. Gesture recognition for virtual reality applications using data gloves and neural networks. In Proceedings of the International Joint Conference on Neural Networks, Washington, DC, USA, 10–16 July 1999; pp. 2043–2046.
6. Yan, S.; Shao, H.D.; Min, Z.S.; Peng, J.J.; Cai, B.P.; Liu, B. FGDAE: A new machinery anomaly detection method towards complex operating conditions. *Reliab. Eng. Syst. Saf.* **2023**, *236*, 109319. [CrossRef]
7. Chen, M.Z.; Shao, H.D.; Dou, H.X.; Li, W.; Liu, B. Data augmentation and intelligent fault diagnosis of planetary gearbox using ILoFGAN under extremely limited samples. *IEEE Trans. Reliab.* **2022**, *72*, 1029–1037. [CrossRef]
8. Wang, Z.J.; Li, Y.J.; Dong, L.; Li, Y.F.; Du, W.H. A RUL Prediction of Bearing Using Fusion Network through Feature Cross Weighting. *Meas. Sci. Technol.* **2023**, *34*, 105908. [CrossRef]
9. Wang, T.Q.; Li, Y.D.; Hu, J.F.; Khan, A.; Liu, L.; Li, C.; Hashmi, A.; Ran, M. A survey on vision-based hand gesture recognition. In *Smart Multimedia*; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2018; Volume 11010, pp. 219–231.
10. Wang, Y.; Zhang, J.; Qin, Y.; Chai, X. Gesture Recognition Method Based on Multi-feature Fusion. *J. Syst. Simul.* **2019**, *31*, 346–352.
11. Oikonomidis, I.; Kyriazis, N.; Argyros, A.; Hoey, J.; McKenna, S. Efficient model-based 3D tracking of hand articulations using Kinect. In Proceedings of the British Machine Vision Conference, Dundee, UK, 29 August–2 September 2011; Volume 2. 11p.
12. Moeslund, T.B.; Hilton, A.; Krüger, V. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.* **2006**, *104*, 90–126. [CrossRef]
13. Yedder, H.B.; Cardoen, B.; Hamarneh, G. Deep learning for biomedical image reconstruction: A survey. *Artif. Intell. Rev.* **2021**, *54*, 215–251. [CrossRef]
14. Suk, H.; Sin, B.; Lee, S. Hand Gesture Recognition Based on Dynamic Bayesian Network Framework. *Pattern Recognit.* **2010**, *43*, 3059–3072. [CrossRef]
15. Bécha Kaâniche, M.; Brémond, F. Gesture Recognition by Learning Local Motion Signatures. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 2745–2752.
16. Baligh, M.; Sabri, A. Arabic online handwriting recognition (AOHR): A survey. *ACM Comput. Surv.* **2017**, *50*, 33.
17. Neidle, C.; Thangali, A.; Sclaroff, S. Challenges in Development of the American Sign Language Lexicon Video Dataset (ASLLVD) Corpus. In Proceedings of the 5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon, LREC 2012, Istanbul, Turkey, 27 May 2012.

18. Camgoz, N.C.; Hadfield, S.; Koller, O.; Ney, H.; Bowden, R. Neural Sign Language Translation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
19. Zhou, H.; Zhou, W.; Qi, W.; Pu, J.; Li, H. Improving Sign Language Translation with Monolingual Data by Sign Back-Translation. In Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021.
20. Zhang, L.G.; Wu, J.Q. Hand Gesture Recognition Based on Hausdorff Distance. *J. Image Graph.* **2002**, *7*, 1144–1150.
21. Yang, X.W.; Feng, Z.Q.; Huang, Z.Z.; He, N.N. Gesture Recognition Based on Combining Main Direction of Gesture and Hausdorff-like Distance. *J. Comput.-Aided Des. Comput. Graph.* **2016**, *28*, 75–81.
22. Zhu, J.Y.; Wang, X.Y.; Wang, W.X.; Dai, G.Z. Hand Gesture Recognition Based on Structure Analysis. *Chin. J. Comput.* **2012**, *29*, 2130–2137.
23. Xin, W.B.; Hao, H.M.; Bu, M.L.; Lan, Y.; Huang, J.H.; Xiong, X.Y. Static gesture real-time recognition method based on ShuffleNetv2-YOLOv3 model. *J. Zhejiang Univ.* **2021**, *55*, 1815–1824.
24. Lin, H.I.; Hsu, M.H.; Chen, W.K. Human hand gesture recognition using a convolution neural network. In Proceedings of the 2014 IEEE International Conference on Automation Science and Engineering, New Taipei, Taiwan, 18–22 August 2014; pp. 1038–1043.
25. Wu, X.Y. A hand gesture recognition algorithm based on DC-CNNJ. *Multimed. Tools Appl.* **2020**, *79*, 91939205.
26. Wu, J.; Tian, Q.; Yue, J. Static gesture recognition based on residual dual attention and cross level feature fusion module. *Comput. Syst. Appl.* **2022**, *31*, 111–119.
27. Jiang, S.; Sun, B.; Wang, L.; Bai, Y.; Li, K.; Fu, Y. Sign Language Recognition via Skeleton-Aware Multi-Model Ensemble. *arXiv* **2021**, arXiv:2110.06161.
28. Papadimitriou, K.; Potamianos, G. Sign Language Recognition via Deformable 3D Convolutions and Modulated Graph Convolutional Networks. In Proceedings of the ICASSP 2023—2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5.
29. Al-onazi, B.B.; Nour, M.K.; Alshahran, H.; Elfaki, M.A.; Alnfiai, 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]
30. Schroff, F.; Kalenichenko, D.; Philbin, J. FaceNet: A Unified Embedding for Face Recognition and Clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; Volume 7, pp. 815–823.
31. 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.
32. 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.
33. Sincan, O.M.; Keles, H.Y. Autsl: A large scale multi-modal turkish sign language dataset and baseline methods. *IEEE Access* **2020**, *8*, 181340–181355. [CrossRef]
34. Selvaraj, P.; Nc, G.; Kumar, P.; Khapra, M. OpenHands: Making Sign Language Recognition Accessible with Pose-based Pretrained Models across Languages. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 22–27 May 2022; Volume 1, pp. 2114–2133.
35. Ryumin, D.; Ivanko, D.; Ryumina, E. Audio-Visual Speech and Gesture Recognition by Sensors of Mobile Devices. *Sensors* **2023**, *23*, 2284. [CrossRef]
36. Jiang, S.; Sun, B.; Wang, L.; Bai, Y.; Li, K.; Fu, Y. Skeleton Aware Multi-modal Sign Language Recognition. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, USA, 19–25 June 2021; pp. 3408–3418.
37. Patra, S.; Maitra, A.; Tiwari, M.; Kumaran, K.; Prabhu, S.; Punyeshwarananda, S.; Samanta, S. Hierarchical Windowed Graph Attention Network and a Large Scale Dataset for Isolated Indian Sign Language Recognition. *arXiv* **2024**, arXiv:2407.14224.
38. Maxim, N.; Leonid, V.; Ruslan, M.; Dmitriy, M.; Iuliia, Z. Fine-tuning of sign language recognition models: A technical report. *arXiv* **2023**, arXiv:2302.07693.
39. Ryumin, D.; Ivanko, D.; Axyonov, A. Cross-Language Transfer Learning Using Visual Information for Automatic Sign Gesture Recognition. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2023**, *XLVIII-2/W3*, 209–216. [CrossRef]
40. De Coster, M.; Van Herreweghe, M.; Dambre, J. Isolated sign recognition from RGB video using pose flow and self-attention. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Nashville, TN, USA, 19–25 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 3441–3450.
41. Zhu, G.; Zhang, L.; Mei, L.; Shao, J.; Song, J.; Shen, P. Large-scale Isolated Gesture Recognition using pyramidal 3Dconvolutional networks. In Proceedings of the International Conference on Pattern Recognition, Cancun, Mexico, 4–8 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 19–24.
42. Wang, P.; Li, W.; Liu, S.; Gao, Z.; Tang, C.; Ogunbona, P. Large-scale isolated gesture recognition using convolutional neural networks. In Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico, 4–8 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 7–12.

43. Li, Y.; Miao, Q.; Tian, K.; Fan, Y.; Xu, X.; Li, R.; Song, J. Large-scale gesture recognition with a fusion of rgb-d data based on the c3d model. In Proceedings of the International Conference on Pattern Recognition, Cancun, Mexico, 4–8 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 25–30.
44. Shen, X.; Zheng, Z.; Yang, Y. StepNet: Spatial-temporal Part-aware Network for Isolated Sign Language Recognition. *ACM Trans. Multimedia Comput. Commun.* **2024**, *226*, 19. [CrossRef]