computers

MDPI

Article

iHand: Hand Recognition-Based Text Input Method for Wearable Devices

Qiang Chu, Chao Ping Chen *, Haiyang Hu, Xiaojun Wu and Baoen Han

Smart Display Lab, Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; cq2491977216@sjtu.edu.cn (Q.C.); hocean@sjtu.edu.cn (H.H.); wu-x-jun@sjtu.edu.cn (X.W.); behan016@sjtu.edu.cn (B.H.)

* Correspondence: ccp@sjtu.edu.cn

Abstract: Text input using hand gestures is an essential component of human–computer interaction technology, providing users with a more natural and enriching interaction experience. Nevertheless, the current gesture input methods have a variety of issues, including a high learning cost for users, poor input performance, and reliance on hardware. To solve these problems and better meet the interaction requirements, a hand recognition-based text input method called iHand is proposed in this paper. In iHand, a two-branch hand recognition algorithm combining a landmark model and a lightweight convolutional neural network is used. The landmark model is used as the backbone network to extract hand landmarks, and then an optimized classification head, which can preserve the space structure of landmarks, is designed to classify gestures. When the landmark model fails to extract hand landmarks, a lightweight convolutional neural network is employed for classification. Regarding the way letters are entered, to reduce the learning cost, the sequence of letters is mapped as a two-dimensional layout, and users can type with seven simple hand gestures. Experimental results on the public datasets show that the proposed hand recognition algorithm achieves high robustness compared to state-of-the-art approaches. Furthermore, we tested the performance of users' initial use of iHand for text input. The results showed that the iHand's average input speed was 5.6 words per minute, with the average input error rate was only 1.79%.

Keywords: wearable device; hand recognition; text input method; human–computer interaction

Citation: Chu, Q.; Chen, C.P.; Hu, H.; Wu, X.; Han, B. iHand: Hand Recognition-Based Text Input Method for Wearable Devices. *Computers* **2024**, *13*, 80. https://doi.org/10.3390/computers13030080

Academic Editor: Lucia Maddalena

Received: 11 February 2024
Revised: 4 March 2024
Accepted: 6 March 2024
Published: 19 March 2024

1. Introduction

Augmented reality and virtual reality are preferably implemented on wearable devices [1–6] for an immersive user experience. Unlike unwearable devices, wearable devices are no longer equipped with the conventional input devices, *e.g.*, keyboards, mice, touch panels, *etc*. In order to input commands and texts in various application scenarios [7–11], speech/voice recognition is a feasible input method. However, speech/voice recognition suffers from many downsides. First, under noisy surroundings, speech recognition is not reliable. Second, for private or sensitive information, *e.g.*, passwords, it is definitely not a good option. Third, users have to passively accept the as-recognized texts. Fourth, for those who cannot talk, the speech/voice recognition is not accessible. For the above scenarios, hand recognition is considered as a better choice.

The hand recognition-based input method simulates the hand movements in human daily life. Compared to physical keyboard input or hand-held controller input, the gesture input method is closer to human habits and actions, providing a more natural and convenient experience. The character encoding module and hand recognition module are the two core modules of the gesture input method. The function of the character encoding module is to encode characters into gestures. The gesture recognition module is responsible for recognizing the gestures made by users. Nowadays, a variety of input methods based on hand gesture recognition have been proposed. However, there are still many problems and challenges to be solved. Firstly, computer vision-based gesture recognition algorithms have

poor robustness due to complex backgrounds [12–14]. Therefore, most of the current input methods have to use gesture recognition algorithms based on hardware devices such as smart gloves, which not only increases the users' cost but also destroys the immersion of the interaction experience. Secondly, in the case of character encoding, whether a character is encoded as one gesture or multiple gestures, there is a high learning cost for users, who either need to learn a large number of gestures or memorize the unfamiliar codes.

In order to solve these problems, we invent a hand gesture text input method called iHand. To avoid the dependence on additional hardware devices such as smart gloves, a computer vision-based gesture recognition algorithm combining a landmark model and a lightweight convolutional neural network model is proposed. The landmark model is used as the backbone network to extract hand landmarks under complex backgrounds, and then an optimized classification head, which can preserve the space structure of landmarks, is designed to classify hand gestures. When the landmark model fails to extract hand landmarks, a lightweight convolutional neural network is used for classification. Thanks to this hand gesture recognition algorithm, iHand only requires the built-in RGB camera of the head-mounted display to recognize gestures and input text, without the need for any additional devices such as controllers. In the character encoding module, to truly reduce learning costs, we encoded characters using a two-dimensional layout mapping approach. The main contributions of this work are given below:

Contribution 1: A robust hand gesture recognition algorithm is designed, providing an efficient and real-time solution for vision-based static hand gesture recognition. The use of the landmark model can greatly improve the robustness of the recognition. The optimized GMLP classification head and the lightweight convolutional neural network can further improve the recognition accuracy.

Contribution 2: A hand recognition-based text input method called iHand is proposed, with which users can input characters by seven simple hand gestures. A user study is conducted to investigate the performance of iHand, proving the proposed technique is efficient and could be adopted for text input.

2. Related Work

In recent years, numerous text input methods have been proposed, such as physical keyboard input, virtual keyboard input, hand gesture input, handwriting, and voice input [15]. Although these methods can input text on wearable devices, they each have their own drawbacks and limitations.

Using a physical keyboard is an easy and simple way to enter text [16–19]. This approach can preserve the user's prior input experience, so it is able to achieve a fast input speed. However, the inability to view the keyboard and hands when wearing a head-mounted display affects input efficiency—particularly for novices who must continually realign their finger placements on the keyboard [16]. Research results show that for inexperienced typists, the inability to view the keyboard and hands leads to a slower input speed and higher error rate. The second issue is that a physical keyboard needs to be placed on a flat surface, which means this method cannot be used in some scenarios, such as when the user is walking.

An additional kind of input method is using a virtual keyboard [20–23]. This method no longer requires a physical keyboard, but instead displays a virtual keyboard in front of the user. To enter characters, users require a way to "press" the keys on the virtual keyboard, such as using controllers [20] or hand gesture recognition [21]. In the CrowbarLimbs system proposed by Bakar *et al.* (Yuan Ze University) [20], users hold controllers with both hands. After moving controllers to the front of the keys, users press the button to input characters. Singhal *et al.* (University of Texas at Dallas) [21] used a marker-based hand tracking system to track the hands of users, in which users make midair clicks to input the characters. Although these methods no longer require physical keyboards, other devices such as smart gloves need to be introduced to press the keys of a virtual keyboard, and users often

need to move their hands over a wide range of space during their interaction with virtual keyboards, which leads to high fatigue and is not conducive to long-term text input.

There are other input methods, such as handwriting input [24], voice input [25], and hand gesture input [26–31]. The study conducted by Venkatakrishnan *et al.* (Clemson University) [24] examined some handwriting input techniques. Although handwriting input has the advantages of a natural interaction and low learning cost, prolonged aerial handwriting can cause high fatigue. In the hand gesture text input method proposed by Sridhar *et al.* (Max Planck Institute for Informatics) [26], each character is encoded as a hand gesture. The learning curve for this method is rather severe, since it requires users to memorize 26 different hand motions. To reduce the quantity of gestures that users need to study, Fallah *et al.* (York University) [28] encoded characters based on Huffman codes. Base-4 Huffman encoding generates non-redundant codes for the English alphabet using the character frequency distribution of a corpus. Although the number of hand gestures is decreased to 4, gestures per character increase to 2.32, meaning that 2.32 gestures are required to enter a character, which slows the speed of text entry. Moreover, users have to memorize Huffman codes, so the total learning cost, including the gestures and codes, is still high. In the works of Fashimpaur *et al.* (Facebook Reality Labs) [30] and Lee *et al.* (Ulsan National Institute of Science and Technology) [31], one hand gesture is associated with multiple characters, not only reducing the number of gestures but also avoiding complex encodings. However, given that multiple letters are associated with a given gesture, a dictionary or language model must be deployed to limit the set of possible words produced by a series of gestures. In addition to the issue of learning cost, most current text input methods based on hand gesture recognition are reliant on additional hardware devices, such as a marker-based hand tracking system, to recognize gestures. The effectiveness of three methods—speech input, gesture input, and physical keyboard input—was compared by Bowman *et al.* (Virginia Polytechnic Institute and State University) [25]. According to the trial results, voice input offers faster and more comfortable input than alternative techniques. Voice input, however, is environment-sensitive and cannot be utilized in certain circumstances, for example, when the user is in a meeting or in a library, it is not suitable to speak. Furthermore, some private information, such as passwords, cannot be spoken in public.

3. Methodology

In this paper, we propose iHand, a hand recognition-based text input method for wearable devices. Figure 1 illustrates the architecture of the proposed iHand, including a hand recognition module, a gesture confirmation module, and a text input module. The role of the hand recognition module is to classify the gesture from the current frame image captured by an RGB camera. As shown in Figure 1, after acquiring the user's image using a normal camera, in order to remove the redundant data in the image, a trained palm detector is used to obtain the bounding box of the hand region. The image inside the bounding box region will be retained, whereas the redundant data outside the region will be removed. Then, the preprocessed hand image is fed into a trained hand landmark model to output 21 hand landmarks, after which we judge the completeness of the hand landmarks. If the quantity of hand landmarks is equal to 21, the hand landmarks are directly input to an optimized classification head containing a message-passing layer to obtain the category label of the hand gesture. When the hand landmarks are incomplete, the original hand image is input to a lightweight convolutional neural network to classify gestures. Through the hand gesture recognition module, the gesture label of the current frame is output. In order to prevent users' misuse or system misrecognition, the gesture confirmation module is designed to detect whether the hand gestures in five consecutive frames are the same or not. If the hand gestures in five consecutive frames are consistent, the module considers the hand gesture valid; otherwise, the hand gesture is invalid. According to the valid hand gesture from the user, the text input module will output the character corresponding to the

gesture. The algorithms of hand recognition and using iHand for text input are shown in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 : Algorithm of hand gesture recognition module

Input: image
Output: class label

```
bounding_box = palm_detector_model(image) // Detect the region of hand
cropped_image = crop_function(bounding_box, image) // Remove redundant data
landmarks = hand_landmark_model(cropped_image) // Get hand landmarks
// If the number of detected hand landmarks is equal to 21, we use GMLP head to classify the
hand gestures. Otherwise, we use a lightweight convolutional neural network for classification.
if the length of landmarks ! = 21 then
  class_label = g_mlp_head(landmarks)
else then
  class_label = cnn_model(image)
end if
return class_label
```

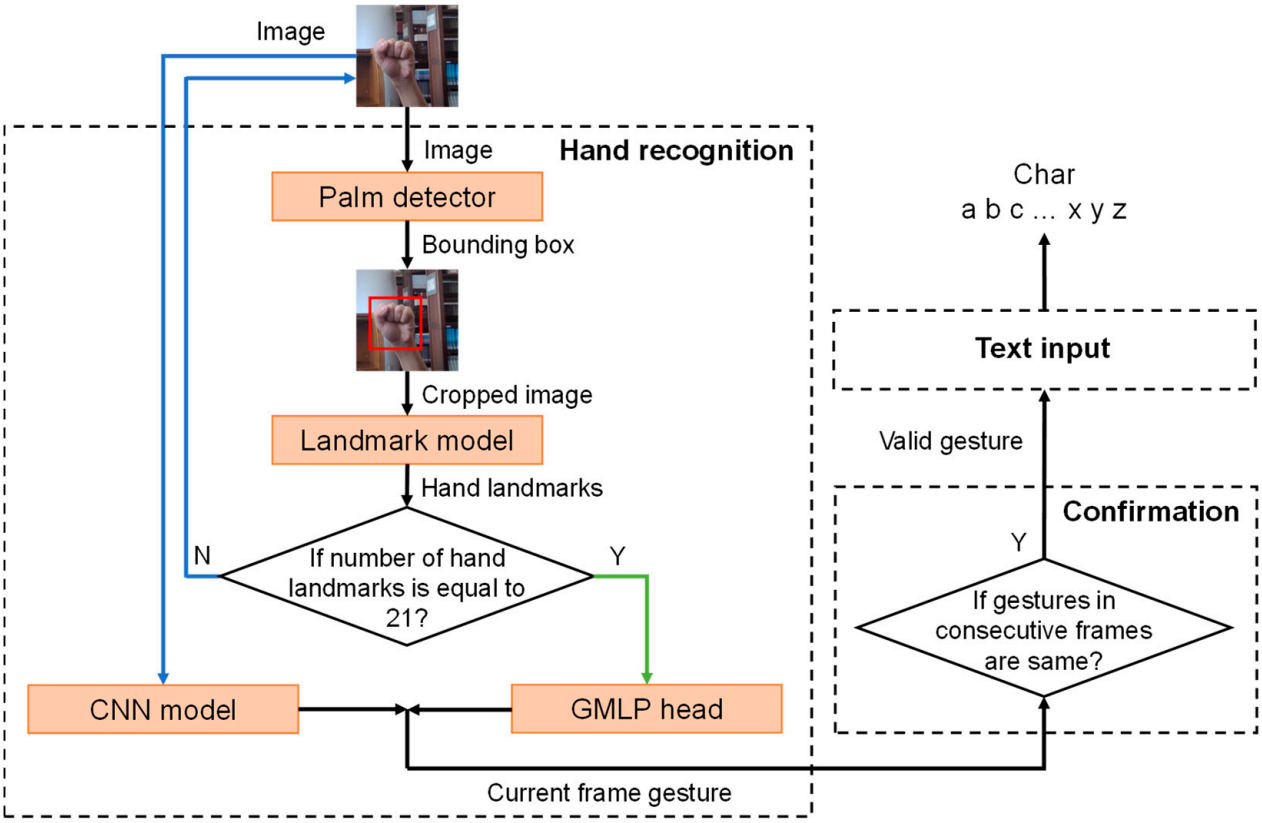

Figure 1. Pipeline of the proposed hand recognition-based text input method called iHand. The hand recognition module outputs class label. The gesture confirmation module detects whether the gesture is valid. The text input module is responsible for entering letters.

Algorithm 2 : Algorithm of using iHand to input characters

Output: character

```
valid_hand_gesture = −1
while true then
  image = camera.get() // Acquire hand images with the camera
  current_frame_gesture = hand_gesture_recognition_module(image)
  if the hand gestures of 5 consecutive frames are the same then
      valid_hand_gesture = current_frame_gesture
  end if
  character = text_input_module(valid_hand_gesture, coding_table)
  break
end while
return character
```

3.1. Hand Recognition Algorithm

We propose a hand gesture recognition algorithm based on the landmark model. Unlike other lightweight convolutional neural network algorithms, our algorithm first uses a landmark model as the backbone network to directionally extract hand joint point features from complex backgrounds and then utilizes a classification head to classify hand gestures based on the joint point features. For the purpose of better classifying gestures based on hand landmarks, the MLP classification head is optimized. A message passing layer is added before the flatten layer, which can transfer information according to the connection relationship between joint points. When using the landmark model to detect joints, the problem of failing to detect complete landmarks may occur. To address this issue, a lightweight network model is employed to classify gestures when the hand landmarks cannot be extracted completely.

In other algorithms, a convolutional neural network model is designed and trained as the backbone network to extract features from the image, and then an MLP classification head is used to classify gestures. Due to the presence of complex backgrounds, the features extracted by the backbone network may come from the background rather than the hand, and thus the robustness of model is bad. To improve the performance, a landmark model that extracts 21 hand landmarks is applied as the backbone network. In this paper, we use the Mediapipe Hands model designed by Google, which can be run in real time to predict the hand landmarks from images taken by a normal RGB camera to acquire the hand landmarks [32]. The Mediapipe Hands model consists of two sub-models: a palm detector model and a hand landmark model. The palm detector model is able to detect and output a hand bounding box. The hand landmark model can predict hand landmarks from the image.

In a convolutional neural network, the model usually uses an MLP classification head after the backbone network to classify gestures. The MLP classification head simply flattens and sums the features extracted from the backbone network by applying a simple weight to them. However, the hand joint points are spatially interconnected and geometrically related. The conventional MLP classification head does not utilize the connectivity between joints when classifying. Therefore, we add a message passing layer before the conventional MLP classification head to build the connectivity between landmarks.

3.1.1. Palm Detector

In order to minimize the effect of redundant information such as the background, before extracting hand joint point coordinates, we use a detector to obtain the hand region and remove the non-hand region in the image. Acquiring hand regions from images is a difficult task because our model needs to detect hands of different shapes and sizes. To address the above challenges, we use a palm detector with a pyramid structure, as shown in Figure 2. Compared to hands with variable shapes, the shape of palms is more fixed, typically a rectangle, and therefore is easier to be detected. The hand area can be

obtained by expanding the palm area. In addition, the network model with an hourglass structure can extract different sizes of bounding boxes, so different scales of hand regions can be detected.

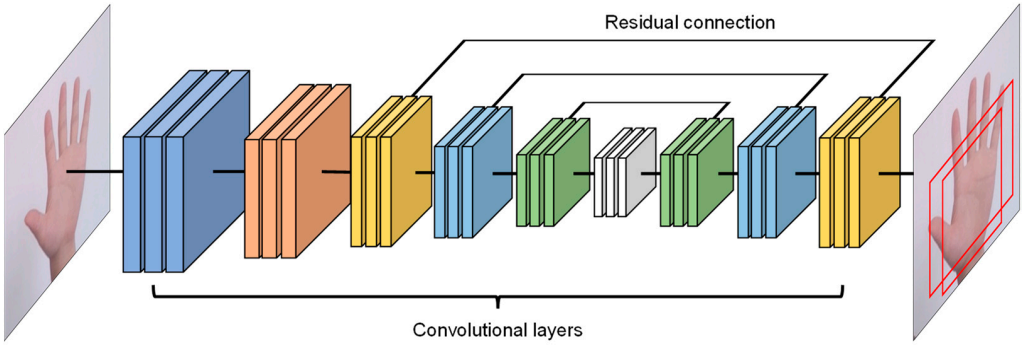

Figure 2. Architecture of the palm detector. The palm detector can easily detect palm regions with a relatively fixed shape. The hand area can be obtained by expanding the palm area. The hourglass-like shape of the model structure helps to detect hands of different scales.

3.1.2. Hand Landmark Model

After obtaining an image of the hand region using the palm detector, as shown in Figure 3, we use a landmark model to extract the coordinates of 21 hand joint points by regression. The landmark model contains three outputs: hand presence, handedness, and hand landmarks. In iHand, we only need the model to output the landmarks of the hand. The input is a picture with height H, width W, and C channels, represented by $x \in \mathbb{R}^{H \times W \times C}$. The output is a sequence of coordinates of the joints, represented by $y \in \mathbb{R}^{N \times 2}$, where N is the number of detected hand joint points, and each joint contains x, y coordinates.

Figure 3. Pipeline of hand landmark model. The input is a picture, and the model has three outputs sharing a feature extractor: hand presence, handedness, and hand landmarks.

3.1.3. GMLP Head

After obtaining the hand joint point features using the landmark model, if the number of detected landmarks is equal to 21, then we use an MLP classification head embedded in a message passing layer, called GMLP, to classify gestures.

The architectures of an MLP head and a GMLP head are shown in Figure 4. Unlike the conventional MLP classification head, the GMLP classification head embeds a message passing layer before the linear layer, allowing each joint point to obtain information about neighboring nodes, which facilitates the subsequent gesture classification. The hand joint points are spatially connected. The conventional MLP head directly flattens the sequence of hand landmarks into a one-dimensional vector and then obtains the output vector by linear transformation, which does not utilize the connectivity between joints, as shown in Figure 5. To better classify gestures, the GMLP first performs a message passing based on the adjacency matrix between the hand joints to build the connectivity between landmarks

and then flattens them to obtain the output vectors through a linear layer. Taking joint 0 as an example: joint 0 is connected to joint 1, 5, 9, 13, and 17. Therefore, the information of nodes 1, 5, 9, 13, and 17 is transmitted to node 0 by adding the vectors of 1, 5, 9, 13, and 17 to the vector of node 0.

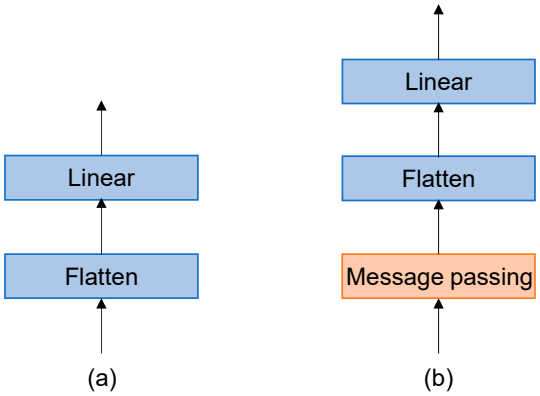

Figure 4. (**a**) Common MLP classification head. (**b**) MLP classification head embedded in the message passing layer called GMLP. GMLP first passes the message based on the adjacency matrix between the hand joints to preserve some of the spatial features and then flattens them to obtain the output vectors through a linear layer.

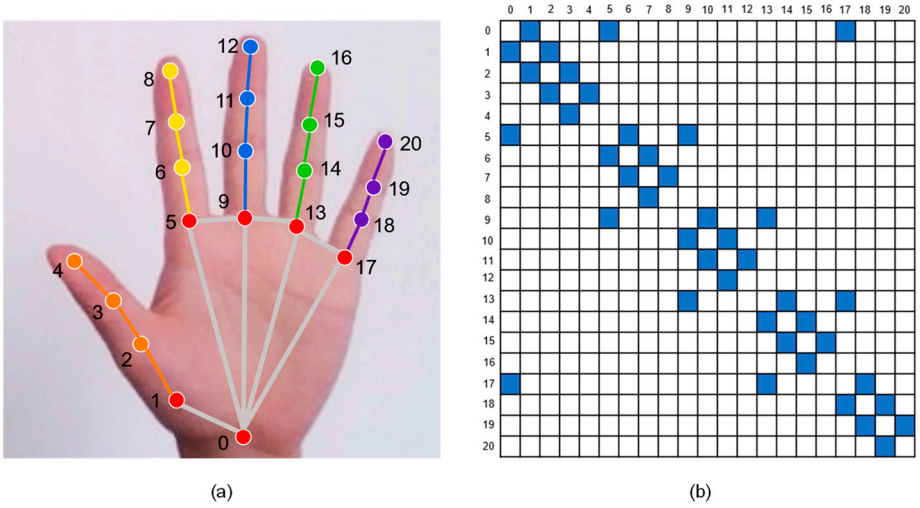

Figure 5. (**a**) Hand landmarks extracted by the model are spatially interconnected to form a skeletal map of hand. (**b**) Adjacency matrix of 21 hand landmarks.

3.1.4. Lightweight Convolutional Neural Network

After obtaining the hand landmarks, if the number of detected articulation points is not equal to 21, then we take the original image as input and use the lightweight convolutional neural network model to obtain the category label. As shown in Figure 6, there are three types of landmark detection result: normal, distorted, and incomplete. When the landmarks are incomplete, we cannot classify gestures through the GMLP head, so a lightweight convolutional neural network model, such as ShuffleNet or MobileNet, is employed to recognize the gesture from the image.

ShuffleNetV2 is a lightweight convolutional neural network model proposed by Megvii for image classification tasks [33]. ShuffleNetV2 uses computationally smaller group convolutions to obtain feature maps and then fuses the feature maps with a shuffling operation. Table 1 presents the overall architecture of ShuffleNetV2. In 2018, Google designed MobileNetV2 using inverted residuals and linear bottlenecks [34], which achieves good results in both image classification tasks and target detection tasks.

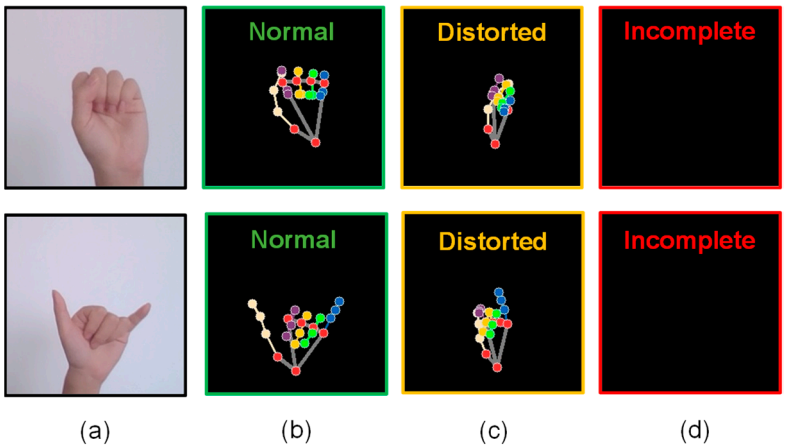

Figure 6. (**a**) Input images. (**b**) Landmark detection results are normal. (**c**) Landmark detection results are distorted. (**d**) Landmark detection results are incomplete.

Table 1. Architecture of ShuffleNetV2. BasicUnit means the basic ShuffleNetV2 unit. Repeat means the number of the operator repeats.

Input	Operator	Output Channels	Stride	Repeat
$224^2 \times 3$	Conv 3×3	24	2	1
$112^2 \times 24$	MaxPool 3×3	24	2	1
$56^2 \times 24$	BasicUnit	116	2	1
$28^2 \times 116$	BasicUnit	116	1	3
$28^2 \times 116$	BasicUnit	232	2	1
$14^2 \times 232$	BasicUnit	232	1	7
$14^2 \times 232$	BasicUnit	464	2	1
$7^2 \times 464$	BasicUnit	464	1	3
$7^2 \times 464$	GlobalPool 7×7	464	/	/
$1^2 \times 464$	Conv 1×1	1024	1	1
$1^2 \times 1024$	Linear	7	/	/

3.2. Alphabet Coding and Input Method

In a hand recognition-based text input method, in addition to the gesture recognition module, the encoding method of letters is also important. The coding method of letters refers to the mapping relationship between the letters and gestures. In iHand, in order to reduce the learning cost for users, we map a one-dimensional sequence of characters into a two-dimensional layout, and the position of a character in the two-dimensional layout is its corresponding codes. Figure 7 shows the two-dimensional character layout and the codes table. Take the letter "m" as an example. The position of the letter "m" in the layout is row 3, column 1, so its corresponding code is 31. Users can use gestures to input letters according to the alphabet codes. In our coding method, users only need to learn 7 simple gestures, as shown in Figure 8, to type the texts.

As shown in Figure 9, take an example of typing the lowercase letter "m". Since the position of "m" in the layout is row 3, column 1, the user first uses gesture 3 to select row 3 and then uses gesture 1 to select column 1 so that the character "m" in row 3, column 1 can be input. In addition to the 26 letters of the alphabet, iHand also supports the "space" and the "delete", which are often used in input tasks. The code for the "space" is 53 and the "delete" is encoded as 54.

	1	2	3	4	5	6
1	a	b	c	d	e	f
2	g	h	i	j	k	l
3	m	n	o	p	q	r
4	s	t	u	v	w	x
5	y	z				

(a)

a 11	b 12	c 13	d 14	e 15	f 16	g 21
h 22	i 23	j 24	k 25	l 26	m 31	n 32
o 33	p 34	q 35	r 36	s 41	t 42	
u 43	v 44	w 45	x 46	y 51	z 52	

(b)

Figure 7. (**a**) Two-dimensional alphabet layout. (**b**) The 2-digit alphabet code table. The 2-digit code of a letter is denoted by the combination of row and column numbers.

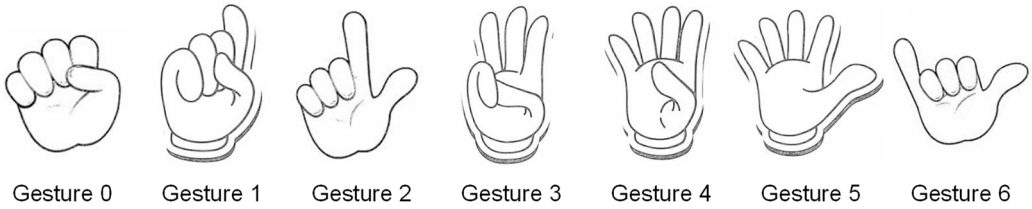

Figure 8. Seven hand gestures used to input letters. Gesture x can select xth row or xth column of the character layout.

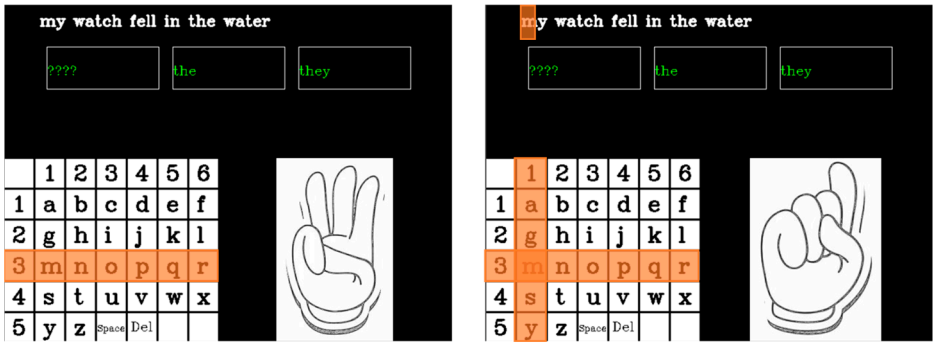

Figure 9. Demonstration of iHand to type letter "m". (**a**) Step 1: gesture 3 to select row 3. (**b**) Step 2: gesture 1 to select column 1. The "m" in row 3, column 1 is entered as highlighted.

4. Hand Recognition

4.1. Datasets and Experimental Environment

The hand gesture recognition algorithm is evaluated on the following public datasets: NUS-II [35] and HGR1 [36]. Table 2 shows some details of NUS-II dataset and HGR1 dataset. The NUS-II dataset contains ten classes of hand gestures from 40 subjects captured under a complex and variable background. The HGR1 dataset consists of 899 images of 27 different sign language gestures performed by 12 individuals. Figure 10 shows some samples from NUS-II and HGR1. To assess the robustness of the proposed algorithm, experiments are conducted on a subject-independent (SI) scheme, which divides images into four folds according to different subjects. When experimenting, three folds are used as the training set, and the remaining fold is used as the test set. This process is repeated four times.

Table 2. Details of NUS-II dataset and HGR1 dataset.

Dataset	Class Number	Subject Number	Image Number
NUS-II [35]	10	40	2000
HGR1 [36]	27	12	899

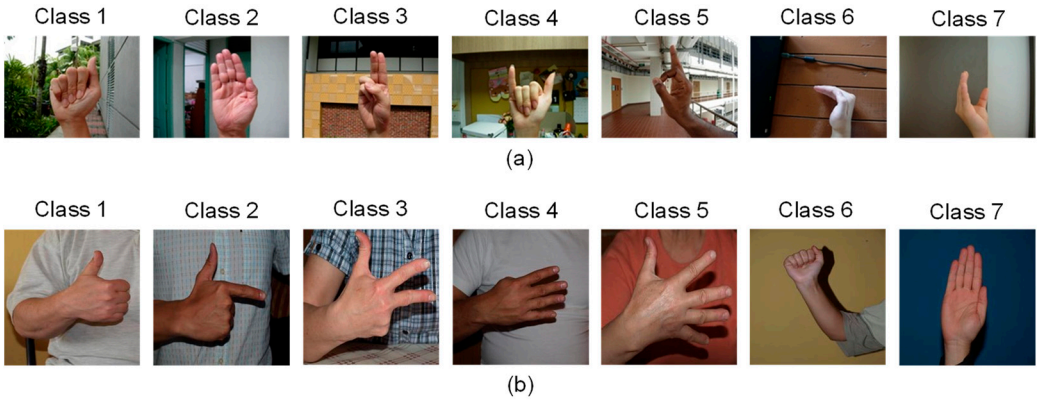

Figure 10. (**a**) RGB images from NUS-II dataset. (**b**) Samples from HGR1 dataset.

The network is trained by NVIDIA Geforce RTX 3060 Laptop GPU. The CPU is i7-11800H, 2.30 GHz with eight cores, and the memory is DDR4 32 GB. The software development environment is CUDA 11.6, Cudnn 8.3.2, and Pytorch 1.13.1.

4.2. Evaluation Criteria

In this paper, Accuracy, Precision, Recall and F1 Score are used as the evaluation metrics of the model. For a binary classification problem that divides instances into positive and negative classes, the prediction results will appear in the following four cases: (1) Predicting the positive class as positive is called TP (True Positive); (2) Predicting the positive class as negative is called FN (False Negative); (3) Predicting the negative class as negative is called TN (True Negative); and (4) Predicting the negative class as positive is called FP (False Positive).

Accuracy refers to the number of instances correctly predicted as a percentage of the total and is calculated using the formula shown below:

$$\text{Accuracy} = (TP + TN)/(TP + FN + TN + FP) \quad (1)$$

Precision refers to the proportion of instances predicted to be in the positive category that are really samples in the positive category and is calculated using the formula shown below:

$$\text{Precision} = TP/(TP + FP) \quad (2)$$

Recall refers to the proportion of samples that are actually positively classified that are predicted to be positively classified and is calculated using the formula shown below:

$$\text{Recall} = TP/(TP + FN) \quad (3)$$

Because in practice, precision and recall are in conflict with each other, the F1 value is used to reconcile and average the precision and recall in order to take into account both precision and recall. The formula for calculating the F1 value is shown below:

$$F1 = 2 \times \text{Precision} \times \text{Recall}/(\text{Precision} + \text{Recall}) \quad (4)$$

4.3. Comparative Experiment

The performance of the proposed recognition algorithm on the NUS-II dataset and HGR1 dataset is summarized in Table 3. Furthermore, a comparative study of the accu-

racy, precision, recall, and F1-score of the proposed network with the existing state-of-art approaches is also given in Tables 4–6. Tables 4 and 5 explicitly state that the proposed approach significantly improves performance in NUS-II and HGR1 datasets, thus establishing the effectiveness of the method proposed in this paper. Table 6 indicates that our algorithm achieves competent results in inference time, with the fastest speed of classification on the HGR1 dataset.

Table 3. Performance of proposed algorithm on NUS-II dataset and HGR1 dataset.

Dataset	Fold	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
NUS-II	1	96.80	96.98	96.80	96.72
	2	98.00	98.05	98.00	97.99
	3	98.20	98.23	98.20	98.20
	4	93.40	93.56	93.40	93.39
	AVG	96.60	96.71	96.60	96.58
HGR1	1	69.95	75.37	71.03	69.24
	2	69.68	71.89	68.56	68.01
	3	79.60	79.99	78.28	78.63
	4	79.92	86.15	79.92	78.59
	AVG	74.79	78.35	74.45	73.62

Table 4. Performance of the state-of-the-art methods and the proposed hand recognition method on the NUS-II dataset.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
ShuffleNetV2 [33]	67.90	69.69	67.90	67.83
MobileNetV2 [34]	78.90	80.65	78.90	78.86
GhostNetV1 [37]	70.60	72.66	70.60	70.71
EfficientNetV2 [38]	67.05	69.76	67.05	66.88
ExtriDeNet [39]	61.49	62.26	61.49	60.60
GhostNetV2 [40]	64.45	66.39	64.45	64.31
Ours	**96.60**	**96.71**	**96.60**	**96.58**

Table 5. Performance of the state-of-the-art methods and the proposed hand recognition method on the HGR1 dataset.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
ShuffleNetV2 [33]	35.73	38.86	35.75	33.07
MobileNetV2 [34]	25.76	27.09	25.84	24.10
GhostNetV1 [37]	34.52	39.94	34.69	33.89
ExtriDeNet [39]	31.15	35.79	31.15	28.86
GhostNetV2 [40]	29.87	38.97	30.13	29.16
Ours	**74.79**	**78.35**	**74.45**	**73.62**

Table 6. Inference time (ms) of the state-of-the-art methods and the proposed method on the NUS-II dataset and HGR1 dataset.

Method	NUS-II	HGR1
ShuffleNetV2 [33]	**4.55**	12.65
MobileNetV2 [34]	10.76	29.19
GhostNetV1 [37]	7.08	18.55
GhostNetV2 [40]	10.66	25.66
Ours	10.46	**10.43**

Table 4 indicates that on the NUS-II dataset containing 2000 images of 10 hand gestures, the highest recognition accuracy of the other algorithms is 78.90%, while our algorithm has a recognition accuracy of 96.60%. In Table 5, compared to the NUS-II, on the more difficult classification task HGR1, which needs to classify 27 hand gestures but just includes 899 samples, the highest recognition accuracy of the other algorithms is only 35.73%, while our algorithm has a recognition accuracy of 74.79%. The resolutions of the images in NUS-II and HGR1 are 120 × 160 and 224 × 224, respectively. Because our algorithm will standardize the image resolution to 256 × 256, the average inference time is relatively close to each other on the two datasets. The other algorithms do not unify the resolution, so they have a faster inference speed on the NUS-II dataset with a smaller resolution and have a slower inference speed on the HGR1 dataset with a bigger resolution.

4.4. Ablation Study

In order to verify the effectiveness of the hand recognition network, ablation studies are conducted in two benchmark datasets: the NUS-II dataset and HGR1 dataset. The results are listed in Tables 7 and 8, respectively. It is apparent that the performance of the network using two branches is better than that of only a landmark model with a classifier head. If there is only a landmark model with a classifier head, when the landmark model fails to extract landmarks, the gesture cannot be classified. The proposed method using two branches, which means that when the landmark model fails to extract landmarks, then the CNN model will step in to recognize hand gestures. The experimental results also indicate that the GMLP head reserving the spatial features of landmarks is better than the MLP head. Compared with the MLP head, the GMLP head utilizes not only the positional information of the points but also the connection information between the points, which helps to improve the accuracy of the recognition.

Table 7. Comparison of proposed algorithm with different settings on NUS-II dataset.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
Landmark model + MLP	93.80	94.27	93.80	93.71
Landmark model + MLP or CNN	95.70	95.93	95.70	95.66
Landmark model + GMLP or CNN	**96.60**	**96.71**	**96.60**	**96.58**

Table 8. Comparison of proposed algorithm with different settings on HGR1 dataset.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)
Landmark model + MLP	72.64	74.60	72.59	71.48
Landmark model + MLP or CNN	73.71	75.67	73.66	72.31
Landmark model + GMLP or CNN	**74.79**	**78.35**	**74.45**	**73.62**

5. Hand Gesture Text Input

The text input experiments are designed to measure the input performance of iHand. Two male and two female participants (average age: 24) from the university are invited to input sentences using iHand. Evaluation metrics such as input speed and input error rate are automatically calculated as the user enters sentences. We also compare the input performance of the iHand and the H4VR [28].

5.1. Experimental Procedure

The user experiment is divided into three phases: the teaching phase, the practice phase, and the testing phase. In the teaching phase, we demonstrate how to use iHand to input characters. In the practice phase, users will independently input five sentences with hand gestures. After the practice phase, the user is required to enter 20 consecutive sentences using iHand. Every five sentences are a block, and there are four blocks in total. The user's input speed on each block will be automatically measured by the system.

The number of incorrect characters typed is also counted to calculate the input error rate. The sentences entered in both the practice and test phases are randomly selected from MacKenzie and Soukoreff's Phrase Sets for evaluating the text input method [41]. The hand fatigue level is given after users input all sentences.

5.2. Evaluation Criteria

The evaluation indexes of the text input method are divided into objective evaluation indexes and subjective evaluation indexes. Objective evaluation indexes include input speed and input error rate. Input speed refers to the number of words that the system can input per minute, and its calculation formula is shown below:

$$\text{WPM} = \text{SSL}/\text{Time} \times 12 \tag{5}$$

where SSL refers to the submitted string length, and Time refers to the time taken to enter the string.

The input error rate refers to the probability that an error occurs when entering a character and is calculated using the formula shown below:

$$\text{CER} = \text{ErrorCharCount}/\text{TotalCharCount} \tag{6}$$

where ErrorCharCount refers to the number of incorrectly entered characters, and TotalCharCount refers to the total number of characters entered.

The subjective evaluation indicator is the degree of hand fatigue, which is categorized into five levels. Level 1 indicates no fatigue and level 5 indicates being very fatigued.

5.3. Text Input Performance

The experimental results are shown in Figure 11. iHand, proposed in this paper, has an average text input speed of 5.6 words per minute, while in H4VR [28], the average text input speed is 4.1 words per minute. The faster input speed of the iHand compared to the H4VR is expected for two reasons: (1) compared with the Huffman encoding method of H4VR, the iHand encoding method allows users to find the corresponding position of characters more quickly, which has a relatively lower learning cost and is easier to use for users in the early stage; and (2) the encoding length of each character in the iHand encoding method is 2 gestures, while the average encoding length of each character in the H4VR encoding method is 2.32 gestures.

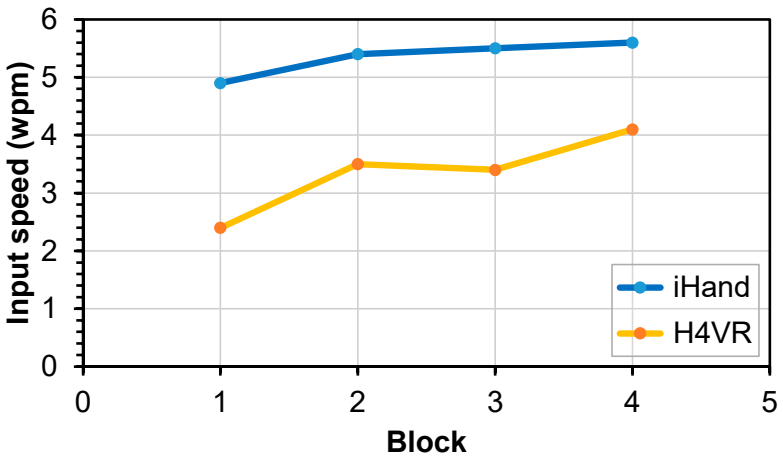

Figure 11. Input speed (words per minute) of iHand and H4VR.

Table 9 presents the input speed, input error rate, and the degree of hand fatigue of H4VR and iHand. The experimental results show that iHand has a faster input speed, lower input error rate, and lower degree of hand fatigue than H4VR, which is in line with our

expectations. Firstly, with the iHand encoding method, users can quickly locate characters for encoding based on their familiarity with the alphabetical order. Secondly, iHand uses only seven hand gestures, and all of these gestures are simpler than that of H4VR, so the learning cost for users is very low. Thirdly, the average encoding length of each character in iHand is shorter than that of H4VR.

Table 9. Input speed, input error rate, and the degree of hand fatigue of H4VR [28] and iHand proposed in this paper.

Method	Input Speed	Input Error Rate	Degree of Hand Fatigue
H4VR [28]	4.1 words per minute	1.82%	2.29
Ours	**5.6** words per minute	**1.79%**	**2.14**

6. Conclusions

In conclusion, we have demonstrated a hand recognition-based text input method called iHand. For the hand recognition algorithm, a landmark model is used as the backbone network to extract hand joint features, and then an optimized classification head is designed to classify hand gestures. When the landmark model fails to extract hand landmarks, a lightweight convolutional neural network model will step in for classification. In iHand, the one-dimensional letter sequence is mapped to a two-dimensional layout, and users can input letters with seven simple hand gestures. The proposed algorithm has been verified on two public datasets: NUS-II and HGR1. In the subject-independent experiments, the hand recognition accuracy of our method reaches 96.90% and 74.79% on NUS-II and HGR1 datasets, respectively, which is significantly higher than that of other similar methods. Overall, the proposed hand gesture recognition algorithm provides a robust, real-time solution for vision-based static hand recognition. Furthermore, in the actual typing tests of iHand, the input speed is 5.6 words per minute, with an input error rate of 1.79% and a hand fatigue of 2.14, which are all better than H4VR hand gesture text input method. In the future, firstly, we will try to use two-handed gesture input or a more efficient character encoding scheme to improve the performance of iHand. Secondly, we will apply the proposed algorithm in other classification tasks, such as sign language recognition. Thirdly, the landmark model will also be optimized to provide more accurate hand landmarks, which helps to improve the recognition accuracy of the algorithm.

7. Software Copyright

In light of the potential application of this work, one software copyright under the name of iHand (certificate number: 11113046) was registered by the National Copyright Administration of China on 9 May 2023.

Author Contributions: Conceptualization, Q.C. and C.P.C.; methodology, Q.C.; software, Q.C.; validation, Q.C. and C.P.C.; formal analysis, Q.C.; investigation, Q.C.; resources, Q.C.; data curation, Q.C.; writing—original draft preparation, Q.C.; writing—review and editing, C.P.C., H.H., X.W. and B.H.; visualization, Q.C. and C.P.C.; supervision, C.P.C.; project administration, C.P.C.; funding acquisition, C.P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by Shanghai Rockers Inc. (15H100000157) and the Natural Science Foundation of Chongqing Municipality (cstb2023nscq-msx0465, cstc2021jcyj-msxmX1136).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare that this study received funding from Shanghai Rockers Inc. The funder was not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication. The authors declare no conflicts of interest.

References

1. Chen, C.P.; Zhou, L.; Ge, J.; Wu, Y.; Mi, L.; Wu, Y.; Yu, B.; Li, Y. Design of retinal projection displays enabling vision correction. *Opt. Express* **2017**, *25*, 28223–28235. [CrossRef]
2. Chen, J.; Mi, L.; Chen, C.P.; Liu, H.; Jiang, J.; Zhang, W. Design of foveated contact lens display for augmented reality. *Opt. Express* **2019**, *27*, 38204–38219. [CrossRef] [PubMed]
3. Chen, C.P.; Mi, L.; Zhang, W.; Ye, J.; Li, G. Waveguide-based near-eye display with dual-channel exit pupil expander. *Displays* **2021**, *67*, 101998. [CrossRef]
4. Chen, C.P.; Cui, Y.; Ye, Y.; Yin, F.; Shao, H.; Lu, Y.; Li, G. Wide-field-of-view near-eye display with dual-channel waveguide. *Photonics* **2021**, *8*, 557. [CrossRef]
5. Chen, C.P.; Cui, Y.; Chen, Y.; Meng, S.; Sun, Y.; Mao, C.; Chu, Q. Near-eye display with a triple-channel waveguide for metaverse. *Opt. Express* **2022**, *30*, 31256–31266. [CrossRef] [PubMed]
6. Chen, C.P.; Ma, X.; Zou, S.P.; Liu, T.; Chu, Q.; Hu, H.; Cui, Y. Quad-channel waveguide-based near-eye display for metaverse. *Displays* **2023**, *81*, 102582. [CrossRef]
7. Innocente, C.; Piazzolla, P.; Ulrich, L.; Moos, S.; Tornincasa, S.; Vezzetti, E. Mixed Reality-Based Support for Total Hip Arthroplasty Assessment. In *Advances on Mechanics, Design Engineering and Manufacturing IV*; Gerbino, S., Lanzotti, A., Martorelli, M., Mirálbes Buil, R., Rizzi, C., Roucoules, L., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 159–169. [CrossRef]
8. Innocente, C.; Ulrich, L.; Moos, S.; Vezzetti, E. A framework study on the use of immersive XR technologies in the cultural heritage domain. *J. Cult. Herit.* **2023**, *62*, 268–283. [CrossRef]
9. Liu, Y.; Fan, X.; Zhou, X.; Liu, M.; Wang, J.; Liu, T. Application of Virtual Reality Technology in Distance Higher Education. In Proceedings of the 2019 4th International Conference on Distance Education and Learning, Shanghai, China, 24–27 May 2019. [CrossRef]
10. Venkatakrishnan, R.; Bhargava, A.; Venkatakrishnan, R.; Lucaites, K.M.; Volonte, M.; Solini, H.; Robb, A.C.; Pagano, C. Towards an Immersive Driving Simulator to Study Factors Related to Cybersickness. In Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Osaka, Japan, 23–27 March 2019. [CrossRef]
11. Thomas, B.H. A survey of visual, mixed, and augmented reality gaming. *CIE* **2012**, *10*, 1–33. [CrossRef]
12. Wang, S.; Zhang, S.; Zhang, X.; Geng, Q. A two-branch hand gesture recognition approach combining atrous convolution and attention mechanism. *Visual Comput.* **2022**, *39*, 4487–4500. [CrossRef]
13. Dadashzadeh, A.; Targhi, A.T.; Tahmasbi, M.; Mirmehdi, M. HGR-Net: A fusion network for hand gesture segmentation and recognition. *IET Comput. Vis.* **2019**, *13*, 700–707. [CrossRef]
14. Alani, A.A.; Cosma, G.; Taherkhani, A.; McGinnity, T.M. Hand gesture recognition using an adapted convolutional neural network with data augmentation. In Proceedings of the 2018 4th International Conference on Information Management (ICIM), Oxford, UK, 25–27 May 2018. [CrossRef]
15. Dube, T.J.; Arif, A.S. Text Entry in Virtual Reality: A Comprehensive Review of the Literature. In *Human-Computer Interaction. Recognition and Interaction Technologies*; Masaaki, K., Ed.; Springer International Publishing: Cham, Switzerland, 2019; pp. 419–437. [CrossRef]
16. Knierim, P.; Schwind, V.; Feit, A.M.; Nieuwenhuizen, F.; Henze, N. Physical Keyboards in Virtual Reality: Analysis of Typing Performance and Effects of Avatar Hands. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal QC, Canada, 21–26 April 2018. [CrossRef]
17. Pham, D.; Stuerzlinger, W. HawKEY: Efficient and Versatile Text Entry for Virtual Reality. In Proceedings of the 25th ACM Symposium on Virtual Reality Software and Technology, Parramatta, NSW, Australia, 12–15 November 2019. [CrossRef]
18. Grubert, J.; Witzani, L.; Ofek, E.; Pahud, M.; Kranz, M.; Kristensson, P.O. Text Entry in Immersive Head-Mounted Display-Based Virtual Reality Using Standard Keyboards. In Proceedings of the 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Reutlingen, Germany, 18–22 March 2018. [CrossRef]
19. Hutama, W.; Harashima, H.; Ishikawa, H.; Manabe, H. HMK: Head-Mounted-Keyboard for Text Input in Virtual or Augmented Reality. In Adjunct Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology, Virtual Event, USA, 10–14 October 2021. [CrossRef]
20. Bakar, M.A.; Tsai, Y.T.; Hsueh, H.H.; Li, E.C. CrowbarLimbs: A fatigue-reducing virtual reality text entry metaphor. *IEEE Trans. Vis. Comput. Graph.* **2023**, *29*, 2806–2815. [CrossRef]
21. Singhal, Y.; Noeske, R.; Bhardwaj, A.; Kim, J.R. Improving Finger Stroke Recognition Rate for Eyes-Free Mid-Air Typing in VR. In Proceedings of the CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April–5 May 2022. [CrossRef]
22. Kern, F.; Niebling, F.; Latoschik, M.E. Text Input for Non-Stationary XR Workspaces: Investigating Tap and Word-Gesture Keyboards in Virtual and Augmented Reality. *IEEE Trans. Vis. Comput. Graph.* **2023**, *29*, 2658–2669. [CrossRef] [PubMed]
23. Speicher, M.; Feit, A.M.; Ziegler, P.; Krüger, A. Selection-based Text Entry in Virtual Reality. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 21–26 April 2018. [CrossRef]
24. Venkatakrishnan, R.; Venkatakrishnan, R.; Chung, C.H.; Wang, Y.S.; Babu, S. Investigating a Combination of Input Modalities, Canvas Geometries, and Inking Triggers on On-Air Handwriting in Virtual Reality. *ACM Trans. Appl. Percept.* **2022**, *19*, 1–19. [CrossRef]

25. Bowman, D.A.; Rhoton, C.J.; Pinho, M.S. Text Input Techniques for Immersive Virtual Environments: An Empirical Comparison. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, Baltimore, MD, USA, 29 September–4 October 2002. [CrossRef]
26. Sridhar, S.; Feit, A.M.; Theobalt, C.; Oulasvirta, A. Investigating the Dexterity of Multi-Finger Input for Mid-Air Text Entry. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, Seoul, Republic of Korea, 18–23 April 2015. [CrossRef]
27. Whitmire, E.; Jain, M.; Jain, D.; Nelson, G.; Karkar, R.; Patel, S.; Goel, M. DigiTouch: Reconfigurable Thumb-to-Finger Input and Text Entry on Head-mounted Displays. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2017**, *1*, 1–21. [CrossRef]
28. Fallah, S.; MacKenzie, S. H4VR: One-handed Gesture-based Text Entry in Virtual Reality Using a Four-key Keyboard. In Proceedings of the Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems, Hamburg, Germany, 23–28 April 2023; Association for Computing Machinery: New York, NY, USA, 2023. [CrossRef]
29. Jiang, H.; Weng, D.; Zhang, Z.; Chen, F. HiFinger: One-Handed Text Entry Technique for Virtual Environments Based on Touches between Fingers. *Sensors* **2019**, *19*, 3063. [CrossRef] [PubMed]
30. Fashimpaur, J.; Kin, K.; Longest, M. PinchType: Text Entry for Virtual and Augmented Reality Using Comfortable Thumb to Fingertip Pinches. In Proceedings of the Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, Honolulu HI, USA, 25–30 April 2020; Association for Computing Machinery: New York, NY, USA, 2020. [CrossRef]
31. Lee, D.; Kim, J.; Oakley, I. FingerText: Exploring and Optimizing Performance for Wearable, Mobile and One-Handed Typing. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021. [CrossRef]
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. [CrossRef]
33. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. *arXiv* **2018**, arXiv:1807.11164. [CrossRef]
34. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv* **2019**, arXiv:1801.04381. [CrossRef]
35. The NUS Hand Posture Dataset-II. Available online: https://scholarbank.nus.edu.sg/handle/10635/137242 (accessed on 3 March 2024).
36. Database for Hand Gesture Recognition. Available online: https://sun.aei.polsl.pl/~mkawulok/gestures/ (accessed on 29 January 2024).
37. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More Features From Cheap Operations. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020. [CrossRef]
38. Tan, M.; Le, Q.V. EfficientNetV2: Smaller Models and Faster Training. *arXiv* **2021**, arXiv:2104.00298. [CrossRef]
39. Bhaumik, G.; Verma, M.; Govil, M.C.; Vipparthi, S.K. ExtriDeNet: An intensive feature extrication deep network for hand gesture recognition. *Visual Comput.* **2022**, *38*, 3853–3866. [CrossRef]
40. Tang, Y.; Han, K.; Guo, J.; Xu, C.; Xu, C.; Wang, Y. GhostNetV2: Enhance Cheap Operation with Long-Range Attention. *arXiv* **2022**, arXiv:2211.12905. [CrossRef]
41. MacKenzie, I.S.; Soukoreff, R.W. Phrase sets for evaluating text entry techniques. In Proceedings of the CHI 2003 Conference on Human Factors in Computing Systems, Fort Lauderdale, FL, USA, 5–10 April 2003. [CrossRef]