applied sciences

MDPI

Article

Ensemble Learning of Multiple Deep CNNs Using Accuracy-Based Weighted Voting for ASL Recognition

Ying Ma [1], Tianpei Xu [1], Seokbung Han [2] and Kangchul Kim [1,*]

[1] Department of Computer Science and Engineering, Chonnam National University, Yeosu 59626, Republic of Korea
[2] Department of Electronic Engineering, Gyeongsang National University, Jinju 52828, Republic of Korea
* Correspondence: kkc@jnu.ac.kr

Abstract: More than four million people worldwide suffer from hearing loss. Recently, new CNNs and deep ensemble-learning technologies have brought promising opportunities to the image-recognition field, so many studies aiming to recognize American Sign Language (ASL) have been conducted to help these people express their thoughts. This paper proposes an ASL Recognition System using Multiple deep CNNs and accuracy-based weighted voting (ARS-MA) composed of three parts: data preprocessing, feature extraction, and classification. Ensemble learning using multiple deep CNNs based on LeNet, AlexNet, VGGNet, GoogleNet, and ResNet were set up for the feature extraction and their results were used to create three new datasets for classification. The proposed accuracy-based weighted voting (AWV) algorithm and four existing machine algorithms were compared for the classification. Two parameters, α and λ, are introduced to increase the accuracy and reduce the testing time in AWV. The experimental results show that the proposed ARS-MA achieved 98.83% and 98.79% accuracy on the ASL Alphabet and ASLA datasets, respectively.

Keywords: ASL; weighted voting; CNN; ensemble learning

Citation: Ma, Y.; Xu, T.; Han, S.; Kim, K. Ensemble Learning of Multiple Deep CNNs Using Accuracy-Based Weighted Voting for ASL Recognition. *Appl. Sci.* **2022**, *12*, 11766. https://doi.org/10.3390/app122211766

Academic Editors: Samuel Morillas, Pedro Latorre-Carmona and Nuria Ortigosa

Received: 18 October 2022
Accepted: 17 November 2022
Published: 19 November 2022

1. Introduction

1.1. Problem Statement

According to the World Health Organization (WHO), approximately 466 million people worldwide suffer from hearing loss, of whom 34 million are children. Moreover, an estimated 900 million people will suffer from hearing loss by 2050 [1]. Sign language is the most effective bridge between the hearing-impaired community and the outside world. Although hearing-impaired people also use written language to communicate, this is inefficient and inconvenient. Sign language uses gestures to articulate meaning in text or speech [2] and sign language recognition uses techniques to recognize these gestures. The emergence of sign language recognition has made the lives of hearing-impaired people more convenient and has eased communication, but most available solutions for sign language recognition are imperfect and inaccurate [3].

Expressions in American Sign Language (ASL) are simple, rich, and diverse [4], but the complex background recognition influence and similarity between gestures are very high. Building a reliable ASL letter recognition model is essential for improving communication as a tool for hearing-impaired people to help spell names and book titles and correct letters. However, complex backgrounds and similarities between gestures make recognition challenging.

Deep learning allows computational models with multiple processing layers to learn and represent data with multiple levels of abstraction, imitating the human brain mechanism and implicitly capturing the complex structures of large-scale data [5]. Deep learning has also enabled better solutions to hundreds of practical problems and has been widely used in natural language processing, human–computer interaction, and other fields. With

the growth of data and the improvement of computing power, the problem of the lack of data and difficulties in training deep neural networks are gradually being solved. At present, computer vision based on deep learning has brought significant progress in image classification [6] and face recognition [7], and the appearance of convolutional neural networks (CNNs) [8] has greatly improved deep learning.

CNNs have different feature extraction structures but a single CNN has difficulty eliminating the influence of the background, so an ensemble of different CNNs is useful for accurate recognition by decreasing the background influence.

1.2. Literature Review

Some machine-learning methods have been used in sign language recognition. Halder et al. [9] proposed a static sign language recognition method based on Principal Component Analysis (PCA) and Support Vector Machine (SVM) to recognize the five vowels in English, with an accuracy of 80%. PCA and SVM were used for hand feature extraction and classification, respectively. Chuan et al. [10] used the Leap Motion sensor to collect 26 letters in the ASL alphabet. The *k*-nearest neighbor algorithm and SVM were used for classification, with accuracies of 72.78% and 79.83%, respectively. Roy et al. [11] used skin-color detection and contour-extraction techniques to detect sign language in videos, and the Camshift and hidden Markov models (HMM) algorithms were used for hand tracking and classification, respectively. Ahmed et al. [12] proposed a skin-color-based detection method to binarize the input data to obtain the face and hand regions, and they calculated the similarity between training and testing data by the dynamic time warping (DTW) algorithm. The DTW algorithm did not use a statistical model framework for training because it was difficult to connect the semantic information of the context. Consequently, DTW had disadvantages in problem solving such as large data volumes and complex gestures. However, most machine-learning methods must convert the original data into feature vectors that are suitable for operation because the image information is too complex and diverse. Thus, the incomplete feature vectors limit the image recognition performance of machine learning methods.

Some deep-learning approaches have been used for sign language recognition. Hasan et al. [13] used a CNN to recognize ASL, achieving 97.62% accuracy. Pigou et al. [14] proposed a CNN-based sign language recognition method to recognize 20 words in the ChaLearn dataset. This model extracted hand and upper-body features from two sets of input data. Jing et al. [15] proposed a multi-channel, multi-modality framework based on a 3D-CNN. The multiple channels contained color, depth, and optical-flow information, and the multiple modalities included gestures, facial expressions, and body poses. Huang et al. [16] proposed a 3D-CNN network model based on multimodal input, including color, depth, and skeleton-joint point information. However, these methods are limited to the single-stream feature extraction portion, and obtaining sufficient feature information to distinguish similar gestures in a single-stream CNN is very difficult due to the high similarity of gestures in ASL.

Gated recurrent unit–relative sign transformer [17], STMC-Transformer [18], and a full transformer network [19] have been successfully used in sign language recognition. The image frame features from sign language videos were extracted and combined into a standard encoder to boost gesture sequence attention and improve recognition performance. However, transformer methods are not appropriate for simple image recognition, such as ASL letters, because these techniques are designed to handle lengthy and complicated sign language videos, so much processing power is wasted.

Some researchers have focused on combining the advantages of different deep-learning models to improve image-recognition performance. Ye et al. [20] proposed a three-dimensional recurrent convolutional neural network (3DRCNN), which combined a 3D-CNN with a fully connected recurrent neural network (FC-RNN). The 3D-CNN learned from color, optical flow, and depth channels. The FC-RNN obtained timing information of sequence segmentation from the video. Yu et al. [21] used deep ensemble learning to decompose body

poses automatically and perceive its background information. Zaidi et al. [22] proposed two methods for automatically constructing ensembles with various architectures, using different architectures to achieve feature diversity. These methods allow image features to be extracted more comprehensively. However, due to overly diverse feature information, the final network layer for decisions tended to ignore small parts of the features.

The self-mutual distillation learning-based system [23] focuses on both short-term and long-term information to enhance the discriminative power for better sign language recognition. 3D ConvNet with the bidirectional long short-term memory system [24] improved sign language recognition performance through data extraction and time-series information. Excellent recognition performance for continuous gestures is achieved by a long short-term memory (LSTM) technique [25] with four different sequences. However, the above methods do not perform well for datasets in which most letters are not continuous gestures.

Transfer deep-learning methods [26,27] have been used to recognize ASL. The advantage of these methods is their ability to fine-tune the weights of the advanced deep neural networks for image recognition to reduce the waste of computational resources while performing well. However, these methods have the slight limitation of incomplete gesture feature extraction. A 2D-CNN with the joint encoding [28] technique performed excellently but had high hardware requirements. A two-stream CNN [29] method used the addition and concatenation operations to extend the feature maps and thus help the CNN better recognize gestures. This method is slightly insufficient in environments with complex backgrounds.

Some researchers are studying ensemble models [30] to improve image recognition performance. A trainable ensemble [31] takes the outputs of individual models to the final decision and demonstrates the possibility of the ensemble being capable of improving sign language recognition performance in learning the correlation of independent model prediction results. Another ensemble-learning method [32] uses various learning algorithms to generate recognition results based on multiple features. Better recognition performance is achieved through a voting scheme. By voting on different recognition results, the models complement their drawbacks, allowing different CNN models to maximize their performance advantages. This allows the CNNs to obtain various features in the feature extraction and reduce the possibility of losing information in the decision portion. Thus, ensemble learning brings new opportunities for better ASL recognition performance.

1.3. Contributions and Structure of the Paper

This paper designs an ASL recognition system for translation applications in the hearing-impaired community, so the goal of this paper is to help hearing-impaired people communicate better with others. The proposed ensemble-learning model for sign language recognition is proposed, using multiple CNNs with accuracy-based weighted voting (AWV) to increase the ASL recognition performance. The contributions of the paper can be summarized as follows:

- An ASL Recognition System using Multiple deep CNNs and Accuracy-based weighted voting (ARS-MA) is proposed, which consists of data preprocessing, feature extraction with multiple deep CNNs, and classification.
- Multiple deep CNNs are designed for feature extraction, and an AWV algorithm is proposed for classification.
- Three new datasets for classification are created from the feature extraction results, with two hyperparameters, α and λ, introduced to optimize the accuracy of the AWV algorithm.
- The proposed model recognizes 29 gestures with accuracies of 98.83% for the ASL Alphabet dataset and 98.79% for the ASLA dataset with complex backgrounds.

The remainder of this paper is organized as follows: Section 2 introduces the details of the methods and models. Section 3 presents the results and comparison of our method to other methods. Finally, Section 4 summarizes the conclusions.

2. Materials and Methods

2.1. Datasets and Image Preprocessing

This paper uses two datasets, the ASL Alphabet [33] and ASLA (American Sign Language Alphabet) [34]. The ASL Alphabet dataset includes 29 classes, comprising 26 alphabetic characters A–Z and three other characters: space, delete, and nothing. Each class has 3000 images, each of which is 200 × 200 pixels in size. The ASLA dataset is similar to the ASL Alphabet except for the backgrounds, as shown in Figure 1. Each class has 7000 images, each of which is 400 × 400 pixels. The two datasets are split into 85% training images and 15% testing data.

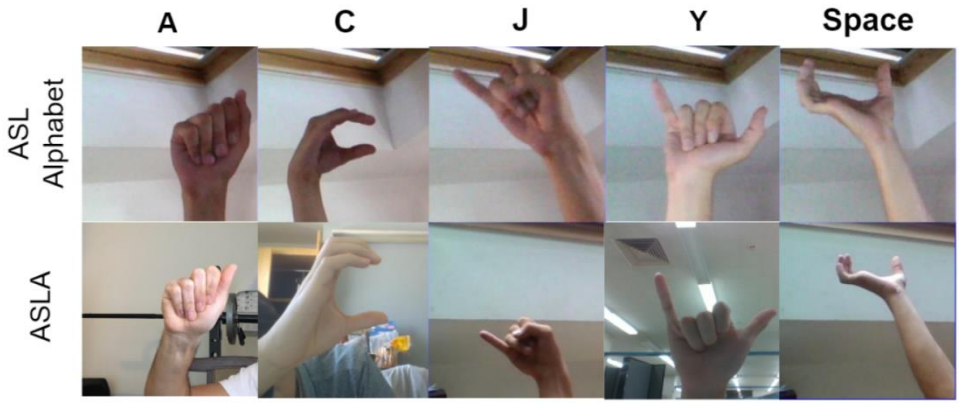

Figure 1. Sample images from the ASL Alphabet and ASLA datasets.

The ASL Alphabet and ASLA datasets have motion, non-motion, and complex background properties, which bring challenges for ASL recognition.

The images are preprocessed using gray data normalization, median filtering, reshaping, and label encoding to reduce the noise and environmental effects. Gray data normalization compresses all pixel data into 0–1 space intervals and changes the image channel to one. Furthermore, it reduces the effects of light in the images and makes them scale invariant, meaning that the mean and variance are the same for all features. The median filter removes background noise from the image. The images are reshaped to 227 × 227 pixels for AlexNet and LeNet and to 224 × 224 pixels for GoogleNet, VGGNet, and ResNet50. Decimal labels are encoded into one-hot vectors to conveniently compare the results in fully connected layers by the label encoding method.

2.2. Proposed Model

An ensemble-learning method allows different CNN models to maximize feature extraction and minimize information loss. Deeper CNN models capture deeper semantic expressions from images but they often ignore information extracted from lower dimensions, losing some image information in the feature extraction process. ARS-MA is proposed to combine the feature capture capabilities of different deep CNN models, as shown in Figure 2. It consists of three steps: data preprocessing, feature extraction and probability prediction by CNN models, and final gesture classification.

CNNs are used for feature extraction and new datasets for different classifiers are created from the results of the multiple deep CNNs. The features extracted from CNNs of different depths increase the diversity of the information to improve recognition performance. The five CNN models are LeNet, AlexNet, GoogleNet, VGGNet, and ResNet, with different feature extraction modes and depths in Step 2, which improve the extraction ability and reduce incomplete semantic expressions. In addition, the five CNN models independently predict probabilities and labels to create new datasets for the next classifier step. The advantage of independent prediction is that it reduces the mutual influence between different depth information in the feature maps.

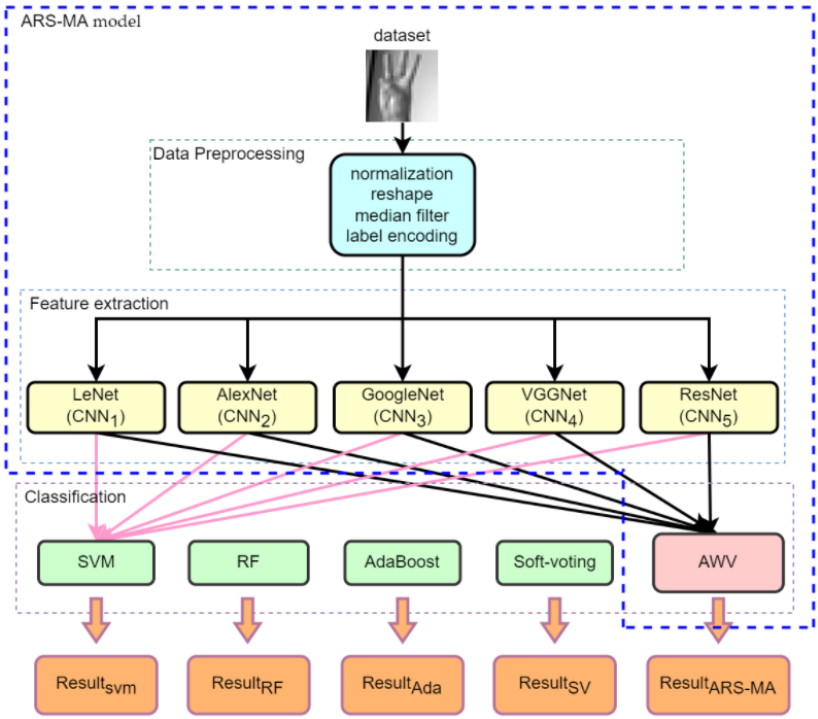

Figure 2. The proposed model schematic.

Although it is relatively difficult for machine-learning algorithms to handle image classification tasks well, they have strong classification abilities for non-image data. SVM, Random Forest (RF), AdaBoost [35], soft voting [36], and the proposed AWV algorithm are used as the final classifier to recognize the ASL alphabet in Step 3.

By comparing the classifiers' recognition performance, the proposed AWV method was selected as the final classifier for the ARS-MA model as shown in the blue box of Figure 2.

2.2.1. New Datasets for Classifiers

Three new datasets (ND1, ND2, and ND3) are built with the results of the multiple deep CNNs for final recognition after they finish the predictions.

Dataset ND1 (P_{CNN1}, P_{CNN2}, P_{CNN3}, P_{CNN4}, P_{CNN5}, $Label_{CNN1}$, $Label_{CNN2}$, $Label_{CNN3}$, $Label_{CNN4}$, $Label_{CNN5}$) becomes the new input data for SVM, RF, and AdaBoost to obtain the final recognition results $Result_{SVM}$, $Result_{RF}$, and $Result_{Ada}$. P_{CNN1}, P_{CNN2}, P_{CNN3}, P_{CNN4}, and P_{CNN5} are the prediction probabilities of the prediction label $Label_{CNN1}$, $Label_{CNN2}$, $Label_{CNN3}$, $Label_{CNN4}$, and $Label_{CNN5}$, respectively, from the five CNN models. For example, $Label_{CNN1}$ is the predicted label from the LeNet model.

Similarly, P_{CNN1} is the set of probabilities of all test images for the predicted class in the LeNet model, calculated by Equation (1):

$$P_{CNN1} = \max(P_{1,1}, P_{1,2}, \ldots, P_{1,29}), \quad (1)$$

where $P_{1,1}$ to $P_{1,29}$ are the probabilities of 29 gesture classes from the output of the LeNet model (CNN1). The max() is defined so as to return the largest value and P_{CNN1} is the largest value among $P_{1,1}$ to $P_{1,29}$.

Dataset ND2 ($P_{1,1}$, $P_{1,2}$, ..., $P_{i,j}$, ..., $P_{5,28}$, $P_{5,29}$) is used for soft voting to obtain the final recognition result of $Result_{SV}$.

Dataset ND3 ($P_{1,1}$, $P_{1,2}$, ..., $P_{i,j}$, ..., $P_{5,28}$, $P_{5,29}$, $ACC_{1,1}$, $ACC_{1,2}$, ..., $ACC_{i,j}$, ..., $ACC_{5,28}$, $ACC_{5,29}$) is used for the AWV algorithm to obtain the final recognition result of $Result_{WDE}$. $P_{i,j}$ and $ACC_{i,j}$ refer to the predicted probability and the accuracy of the jth gesture class in the ith CNN. For example, $ACC_{1,1}$ is the accuracy gained after finishing the

training process for the LeNet model in Class A; all test images of Class A have the same $ACC_{1,1}$ in the next voting step, and $P_{1,1}$ is the probability of all test images for Class A in the LeNet model.

2.2.2. Classification and the Proposed AWV Algorithm

The final classifier in the ARS-MA model was selected from among five algorithms: SVM, RF, AdaBoost [35], soft voting, and the AWV algorithm. These obtained final results from the aspects of four classification methods: nonlinear mapping (SVM), an ensemble tree (RF), a weighted ensemble tree (AdaBoost), and voting (soft-voting and the AWV algorithm).

The keys of the SVM algorithm establish the maximum margin hyperplane for classification and have a good generalization ability. The RF algorithm creates new training samples, which are used to train several different decision trees. Then, the final result is aggregated from the various decision trees. Boosting is a machine-learning approach that aims to create a highly accurate model by combining many less accurate models. AdaBoost, the most widely used boosting algorithm, combines many decision trees and gives greater weights to the higher-accuracy tree classifiers. The new dataset ND1 is used for SVM, RF, and AdaBoost to fuse the CNNs.

In soft voting, the probabilities of five CNN models for each class are averaged to predict the result. The new dataset ND2 is used for the soft-voting algorithm, which compares the 29 average probabilities from the five CNN models for each class, that is, the comparison of $[(P_{1,1} + P_{2,1} + P_{3,1} + P_{4,1} + P_{5,1})/5, \ldots, (P_{1,29} + P_{2,29} + P_{3,29} + P_{4,29} + P_{5,29})/5]$.

However, a class result having a high probability from a CNN model does not mean that the class is correct. For example, AlexNet performs better than LeNet in general. When both are used in an ensemble together, LeNet incorrectly predicts Class A gestures into other classes, while AlexNet accurately predicts them in Class A, but the predicted probability of Class A in LeNet is coincidentally higher than in AlexNet. In this case, soft voting only considers the probability, which makes it difficult to help the ensemble model correct the error. Soft voting makes the final recognition of the gesture class result with the highest prediction probability but does not always consider different CNN models. Therefore, a new method AWV is proposed to consider accuracy when voting for the gesture class. This method is called Accuracy-based Weighted Voting (AWV).

The soft-voting algorithm only focuses on the probabilities for the final recognition of the ensemble model, but the AWV algorithm considers the CNN prediction accuracies and the probabilities corresponding to each class, which allows the ensemble model to fuse the results of the CNNs more accurately. The new dataset ND3 is used for the AWV algorithm. The weights in the proposed AWV algorithm are defined in Equation (2):

$$w_{i,j} = \frac{{ACC_{i,j}}^{\alpha}}{\lambda}, \tag{2}$$

where $w_{i,j}$ is the weight value of the jth class of the dataset in the ith CNN classifier; $ACC_{i,j}$ is the recognition accuracy of the jth class in the ith CNN classifier; and α is an arbitrary number. The accuracy of each CNN is less than one, and ${ACC_{i,j}}^{\alpha}$ increases the difference between $w_{i,j}$. λ is an arbitrary number to reduce the testing time and its value is assigned by simulation to prevent the weights from becoming too large.

The output of the AWV algorithm is defined as follows in Equation (3):

$$\text{Output} = \text{Label}(\max(\frac{\sum_{i=1}^{N} w_{i,1} * P_{i,2}}{N}, \ldots, \frac{\sum_{i=1}^{N} w_{i,j} * P_{i,j}}{N}, \ldots, \frac{\sum_{i=1}^{N} w_{i,29} * P_{i,29}}{N})), \tag{3}$$

where $P_{i,j}$ is the probability calculated by the j_{th} class in the i_{th} CNN classifier. Each CNN model calculates the probabilities for each class, and these probabilities are combined with weights to calculate a group of values using the formula $\frac{\sum_{i=1}^{N} w_{i,j} * P_{i,j}}{N}$. The largest value is

chosen by the max() function. Label() is defined to take the class number belonging to the largest value as the final gesture class prediction result, meaning the final ASL letter recognition result is determined by comparing 29 values: $[w_{1,1} * P_{1,1} + w_{2,1} * P_{2,1} + w_{3,1} * P_{3,1} + w_{4,1} * P_{4,1} + w_{5,1} * P_{5,1})/5, \ldots, (w_{1,29} * P_{1,29} + w_{2,29} * P_{2,29} + w_{3,29} * P_{3,29} + w_{4,29} * P_{4,29} + w_{5,29} * P_{5,29})/5]$. The workflow of the AWV algorithm is shown in Figure 3.

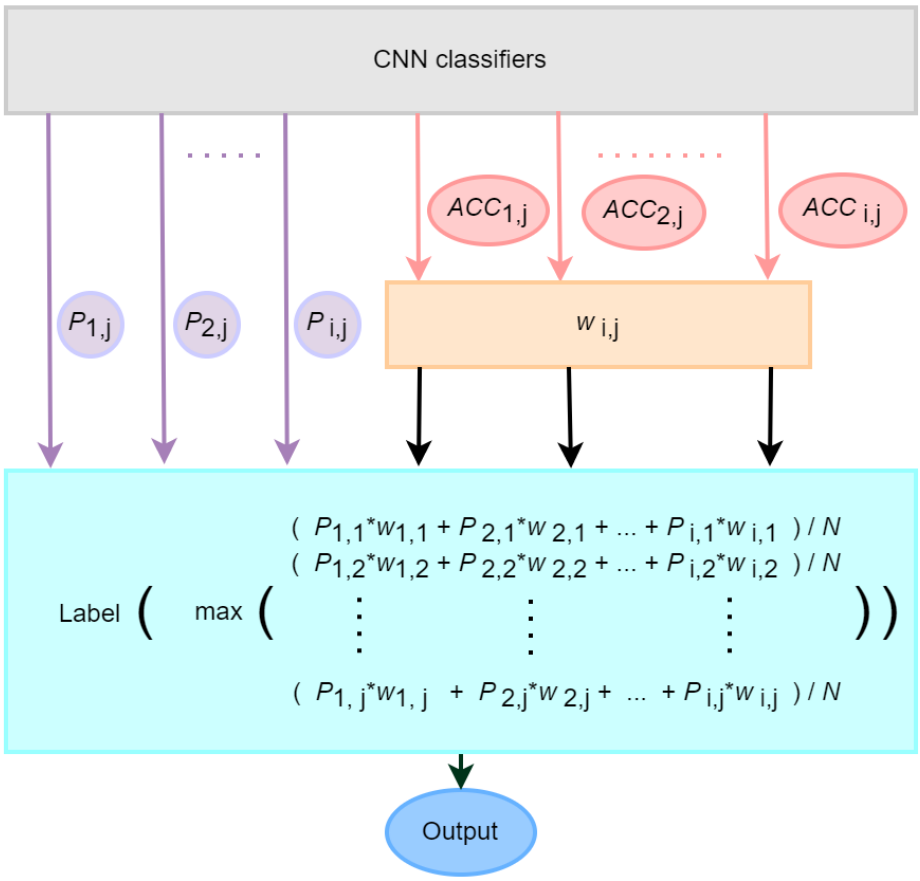

Figure 3. The process of the proposed accuracy-based weighted voting (AWV) algorithm.

In the AWV algorithm, the five probabilities for each sign language class are predicted by five different CNN models, and the weights of each class are determined by the accuracy of each corresponding CNN model. The total number of weights is 5×29 in the ARS-MA model, recognizing 29 gesture classes in the ASL Alphabet and ASLA datasets. AWV makes the CNN models better complement their drawbacks after independent probability predictions. The execution process of the AWV algorithm with $\alpha = 2$, $\lambda = 5$ is shown in Algorithm 1. The proposed weighted voting is accuracy-based, where a CNN model with higher recognition accuracy is given greater weight in the AWV to improve the accuracy of the ARS-MA model.

Algorithm 1. The AWV algorithm for ASL letter recognition ($\alpha = 2$, $\lambda = 5$).

Input:
$X = [P_{1,1}, P_{1,2}, \ldots, P_{i,j}, \ldots, P_{5,28}, P_{5,29}, ACC_{1,1}, ACC_{1,2}, \ldots, ACC_{i,j}, \ldots, ACC_{5,28}, ACC_{5,29}]$
Output: Y
Process:
1) *Initialize the weights matrix for memory*
weights = $\mathbf{W}[w_{i,j}]$
$w_{i,j}$ = 1 for i = 1, 2, . . . , 5 and for j = 1, 2, . . . , 29

2) *Calculate weights*
For i from 1 to 5 **do**
For j from 1 to 29 **do**

$$w_{i,j} = \frac{ACC_{i,j}{}^2}{5}$$

weights ← $w_{i,j}$
3) *Initialize the values matrix for memory*
values = **V**[$value_j$]
$value_j$ = 1 for j = 1, 2, . . . , 29
4) *Calculate a total of 29 values for the final decision*
For j from 1 to 29 **do**
$value_j = (w_{1,j} \times P_{1,j} + w_{2,j} \times P_{2,j} + w_{3,j} \times P_{3,j} + w_{4,j} \times P_{4,j} + w_{5,j} \times P_{5,j})$ / 5
values ← $value_j$
5) *Sort a total of 29 values from largest to smallest*
For j from 1 to 29 **do**
sorted_values = largest_to_smallest_sort(values)
6) *Take the class j belonging to the largest value*
Largest value = sorted_values.take(0) = $value_j$ *//take the first largest value*
Y ← *j*

2.2.3. CNN Algorithms

The main structures in LeNet [37], AlexNet [38], and VGGNet [39] are shown in Figure 4a–c. The inception structure of GoogleNet [40], whose width increases by branching and merging to improve the accuracy of the model, is shown in Figure 4d. The residual module of ResNet [41] in Figure 4e generates an output F(x) + x from an input x and reduces the gradient-vanishing problem, improving the model performance in the prediction task.

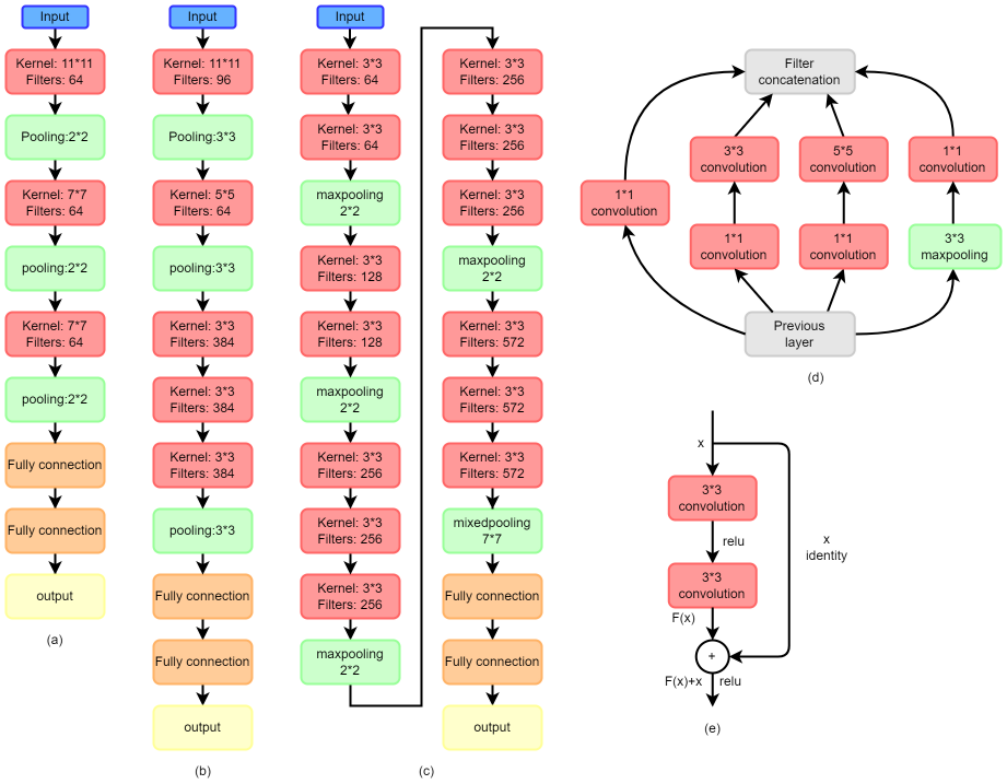

Figure 4. The structure of the convolutional neural network (CNN) methods: (**a**) LeNet5, (**b**) AlexNet, (**c**) VGGNet, (**d**) the inception structure in GoogleNet, and (**e**) the residual module in ResNet.

Five M-CNN models are designed based on the above models: M-LeNet, M-AlexNet, M-GoogleNet, M-VGGNet, and M-ResNet.

Table 1 shows the structure of M-LeNet in detail. The size of the kernels is modified, and one convolution layer is reduced to decrease the consumption time. Table 2 shows the structure of M-AlexNet. Batch normalization [42] is added after each convolution layer in M-AlexNet to reduce the impact of unstable gradients. Table 3 shows the structure of M-GoogleNet in detail, where MP indicates a max-pooling layer, the convolution stride is one, and there is one MP layer in the inception block. Table 4 shows the structure of M-VGGNet in detail, where the 3 × 3 convolution kernel and 2 × 2 max-pooling size are used in the entire network to improve the performance by continuously deepening the network structure. Batch normalization is added behind each convolution layer. The layer configuration in M-VGGNet is similar to LeNet and AlexNet. The purpose of using M-LeNet, M-AlexNet, and M-VGGNet is to help the ARS-MA model extract gesture semantic information at different depths with a similar layer configuration in the feature extraction step. Table 5 shows the structure of M-ResNet in detail, where the number of convolution layers is reduced to obtain better performance for the ASL Alphabet and ASLA datasets.

Table 1. The architecture of M-LeNet5.

CNN Model	Architecture	Kernel Size	Output Shape
M-LeNet	Input	-	227 × 227 × 1
	Convolution_1	5 × 5, 64, stride = 2	75 × 75 × 64
	Maxpooling_1	2 × 2, stride = 2	37 × 37 × 64
	Convolution_2	5 × 5, 128, stride = 2	11 × 11 × 128
	Maxpooling_2	2 × 2, stride = 2	5 × 5 × 128
	Flatten	-	3200
	Fully connected_1	-	640
	Fully connected_2	-	128
	Output	-	29

Table 2. The architecture of M-AlexNet.

CNN Model	Architecture	Kernel Size	Output Shape
M-AlexNet	Input	-	227 × 227 × 1
	Convolution_1	11 × 11, 64, stride = 4	55 × 55 × 64
	Maxpooling_1	3 × 3, stride = 2	27 × 27 × 64
	Convolution_2	5 × 5, 128, stride = 1	27 × 27 × 128
	Maxpooling_2	3 × 3, stride = 2	13 × 13 × 128
	Convolution_3	3 × 3, 256, stride = 1	13 × 13 × 256
	Convolution_4	3 × 3, 256, stride = 1	13 × 13 × 256
	Convolution_5	3 × 3, 128, stride = 1	13 × 13 × 128
	Maxpooling_5	3 × 3, stride = 2	6 × 6 × 128
	Flatten	-	4600
	Fully connected_1	-	2096
	Fully connected_2	-	2096
	Output	-	29

M-GoogleNet, M-VGGNet, and M-ResNet are all high-depth models with different layer configurations. Hence, the ARS-MA model gains the ability to obtain more high-depth information from the feature extraction because the gesture semantic information is easily captured in the high-depth feature maps [43] and deeper feature information categories are obtained. In all models, dropout is used to reduce overfitting, the activation function in the convolution layers is the rectified linear unit (ReLu) function, and the softmax function is used for multi-class prediction.

Table 3. The architecture of M-GoogleNet.

CNN Model	Architecture	Kernel Size			Output Shape
M-GoogleNet	Input	-			224 × 224 × 1
	Convolution_1	7 × 7, 64, stride = 2			112 × 112 × 64
	Maxpooling_1	3 × 3, 64, stride = 2			56 × 56 × 64
	Convolution_2	3 × 3, 192, stride = 1			56 × 56 × 192
	Maxpooling_2	3 × 3, 64, stride = 2			28 × 28 × 192
	Inception_1	1 × 1, 64	1 × 1, 96 3 × 3, 128	1 × 1, 32 5 × 5, 32	3 × 3, 32 (MP) 1 × 1, 32	28 × 28 × 256
	Inception_2	1 × 1, 128	1 × 1, 128 3 × 3, 192	1 × 1, 32 5 × 5, 96	3 × 3, 64 (MP) 1 × 1, 64	28 × 28 × 480
	Maxpooling_3	3*3, stride = 2				14 × 14 × 480
	Inception_3	1 × 1, 192	1 × 1, 96 3 × 3, 208	1 × 1, 16 5 × 5, 48	3 × 3, 64 (MP) 1 × 1, 64	14 × 14 × 512
	Inception_4	1 × 1, 160	1 × 1, 112 3 × 3, 224	1 × 1, 24 5 × 5, 64	3 × 3, 64 (MP) 1 × 1, 64	14 × 14 × 512
	Inception_5	1 × 1, 128	1 × 1, 128 3 × 3, 256	1 × 1, 24 5 × 5, 64	3 × 3, 64 (MP) 1 × 1, 64	14 × 14 × 512
	Inception_6	1 × 1, 112	1 × 1, 144 3 × 3, 288	1 × 1, 32 5 × 5, 64	3 × 3, 64 (MP) 1 × 1, 64	14 × 14 × 528
	Inception_7	1 × 1, 256	1 × 1, 160 3 × 3, 320	1 × 1, 32 5 × 5, 128	3 × 3, 128 (MP) 1 × 1, 128	14 × 14 × 832
	Maxpooling_4	3 × 3, stride = 2				7 × 7 × 832
	Inception_8	1 × 1, 256	1 × 1, 160 3 × 3, 320	1 × 1, 32 5 × 5, 128	3 × 3, 128 (MP) 1 × 1, 128	7 × 7 × 832
	Inception_9	1 × 1, 384	1 × 1, 192 3 × 3, 384	1 × 1, 48 5 × 5, 128	3 × 3, 128 (MP) 1 × 1, 128	7 × 7 × 1024
	Avgpooling_1	7 × 7, 1024, stride = 1				1 × 1 × 1024
	Fully connected_1	-				1 × 1 × 1000
	Output	-				29

Table 4. The architecture of M-VGGNet.

CNN Model	Architecture	Kernel Size	Output Shape
M-VGGNet	Input	-	224 × 224 × 1
	Convolution_1	3 × 3, 64, stride = 2	112 × 112 × 64
	Convolution_2	3 × 3, 64, stride = 1	112 × 112 × 64
	Maxpooling_1	2 × 2, stride = 2	56 × 56 × 64
	Convolution_3	3 × 3, 128, stride = 1	56 × 56 × 128
	Convolution_4	3 × 3, 128, stride = 1	56 × 56 × 128
	Maxpooling_2	2 × 2, stride = 2	28 × 28 × 128
	Convolution_5	3 × 3, 256, stride = 1	28 × 28 × 256
	Convolution_6	3 × 3, 256, stride = 1	28 × 28 × 256
	Convolution_7	3 × 3, 256, stride = 1	28 × 28 × 256
	Maxpooling_3	2 × 2, stride = 2	12 × 12 × 256
	Convolution_8	3 × 3, 512, stride = 1	12 × 12 × 512
	Convolution_9	3 × 3, 512, stride = 1	12 × 12 × 512
	Convolution_10	3 × 3, 512, stride = 1	12 × 12 × 512
	Maxpooling_4	2 × 2, stride = 2	6 × 6 × 512

Table 4. *Cont.*

CNN Model	Architecture	Kernel Size	Output Shape
	Convolution_11	3 × 3, 512, stride = 1	6 × 6 × 512
	Convolution_12	3 × 3, 512, stride = 1	6 × 6 × 512
	Convolution_13	3 × 3, 512, stride = 1	6 × 6 × 512
	Maxpooling_5	3 × 3, stride = 2	3 × 3 × 512
	Flatten	-	4600
	Fully connected_1	-	4096
	Fully connected_2	-	1000
	Output	-	29

Table 5. The architecture of M-ResNet.

CNN Model	Architecture	Kernel Size	Output Shape
M-ResNet	Input	-	224 × 224 × 1
	Convolution_1	7 × 7, 64, stride = 2	112 × 112 × 64
	Maxpooling_1	3 × 3, stride = 2	56 × 56 × 64
	Residual module_1 ×2	3 × 3, 128, stride = 13 × 3, 128, stride = 1	56 × 56 × 128
	Residual module_2 ×1	3 × 3, 256, stride = 23 × 3, 256, stride = 1	28 × 28 × 256
	Residual module_3 ×1	3 × 3, 256, stride = 13 × 3, 256, stride = 1	28 × 28 × 256
	Residual module_4 ×1	3 × 3, 256, stride = 23 × 3, 256, stride = 1	14 × 14 × 256
	Residual module_5 ×1	3 × 3, 256, stride = 13 × 3, 256, stride = 1	14 × 14 × 256
	Residual module_6 ×1	3 × 3, 512, stride = 23 × 3, 512, stride = 1	7 × 7 × 512
	Residual module_7 ×1	3 × 3, 512, stride = 13 × 3, 512, stride = 1	7 × 7 × 512
	Avgpooling_1	7 × 7, stride = 1	1 × 1 × 512
	Fully connected_1	-	512
	Output	-	29

2.3. Evaluation Methods

The performances of the five M-CNNs were evaluated after completing the training. The performance scores [44] were compared using the receiver operating characteristics (ROC) curve, area under the ROC curve (AUC), accuracy, recall, precision, and F1-score evaluation methods. The ROC curve is a tool to examine the classifier performance. AUC is an important metric for model comparison calculated from the ROC. The ROC curve for a multi-class problem is obtained by averaging the ROC curves of each class.

The accuracy, recall, precision, and F1-score are calculated as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

$$\text{F1} - \text{score} = \frac{2\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{7}$$

where TP is true positive, FN is false negative, FP is false positive, and TN is true negative. These four evaluation methods are used to measure the effectiveness of the model.

3. Results and Analysis

The performance results for the five M-CNN models are shown in Table 6. The accuracies for all M-CNNs are higher than 93%, which means that each M-CNN model effectively extracts useful features for gesture prediction in ASL letter images. The performance scores of the M-CNNs decrease in the ASLA dataset, which has complex background environments, but they are no more than 1% lower than the ASL Alphabet dataset, which proves that the M-CNN models achieve almost the same results on both datasets. M-ResNet with the residual module had the best recognition performance for both datasets of all five M-CNNs because high-depth models with high-depth feature maps are well suited to extracting semantic information for prediction.

Table 6. Performance evaluation of the five CNNs on both datasets.

CNN Methods	ASL Alphabet				ASLA			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
M-LeNet5	94.45%	94.76%	94.67%	94.71%	93.53%	95.11%	94.27%	94.69%
M-AlexNet	95.56%	94.46%	94.38%	94.42%	94.61%	94.11%	93.87%	93.99%
M-GoogleNet	96.07%	91.84%	90.41%	91.12%	95.54%	90.87%	90.92%	90.89%
M-VGGNet	97.39%	93.48%	93.74%	93.61%	96.86%	93.78%	92.69%	93.18%
M-ResNet	97.76%	95.42%	96.13%	95.77%	97.27%	95.61%	96.33%	95.97%

The recognition scores of the five different classifiers in the classification step are shown in Table 7. The accuracy, precision, recall, and F1-scores range from 97.89% to 98.83%, 96.12% to 97.98%, 95.78% to 97.59%, and 95.95% to 97.60%, respectively. Comparing Tables 7 and 8 shows that the classification accuracies of the fusion classifiers are higher than those in a single CNN, proving that the proposed ensemble models using multiple deep CNNs are superior to single CNN models.

Table 7. Performance evaluation of five classifiers on two datasets.

Final Fusion Classifiers	ASL Alphabet				ASLA			
	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
SVM	98.35%	97.67%	97.12%	97.39%	98.27%	97.59%	97.31%	97.45%
RF	97.96%	96.32%	96.27%	96.29%	97.89%	97.56%	96.22%	96.89%
AdaBoost	98.11%	96.12%	95.78%	95.95%	98.04%	96.24%	96.51%	96.37%
Soft voting	98.46%	97.19%	96.51%	96.85%	98.41%	96.74%	96.84%	96.79%
AWV (α = 2, λ = 5)	98.83%	97.98%	96.94%	97.46%	98.79%	97.61%	97.59%	97.60%

Table 8. Accuracies by α in ARS-MA model.

α	Accuracy on ASL Alphabet	Accuracy on ASLA
1	98.7843%	98.7432%
2	98.8319%	98.7943%
3	98.8317%	98.7937%
4	98.8317%	98.7942%

The recognition accuracies for the two datasets in Table 7 are nearly the same, proving that the proposed ARS-MA effectively removes the noise in a complex background. The proposed AWV model was the most accurate, at 98.83% and 98.79% for the ASL Alphabet and ASLA datasets, respectively, so it was chosen as the classifier in the classification step.

The ROC curves and AUC values of the five M-CNNs for the two datasets are shown in Figure 5a,b, respectively; M-ResNet achieved the highest AUCs of 0.89 on the ASL Alphabet dataset and 0.88 on the ASLA dataset. The ROC curves and the AUC values of

the five classifiers for the two datasets are depicted in Figure 5c,d. The AWV algorithm achieved the highest AUCs, of 0.93 and 0.91, on the ASL Alphabet and ASLA datasets, respectively. The AUCs of all single M-CNNs were lower than that of AWV in the ARS-MA model, which proves the proposed ARS-MA model works well for both datasets.

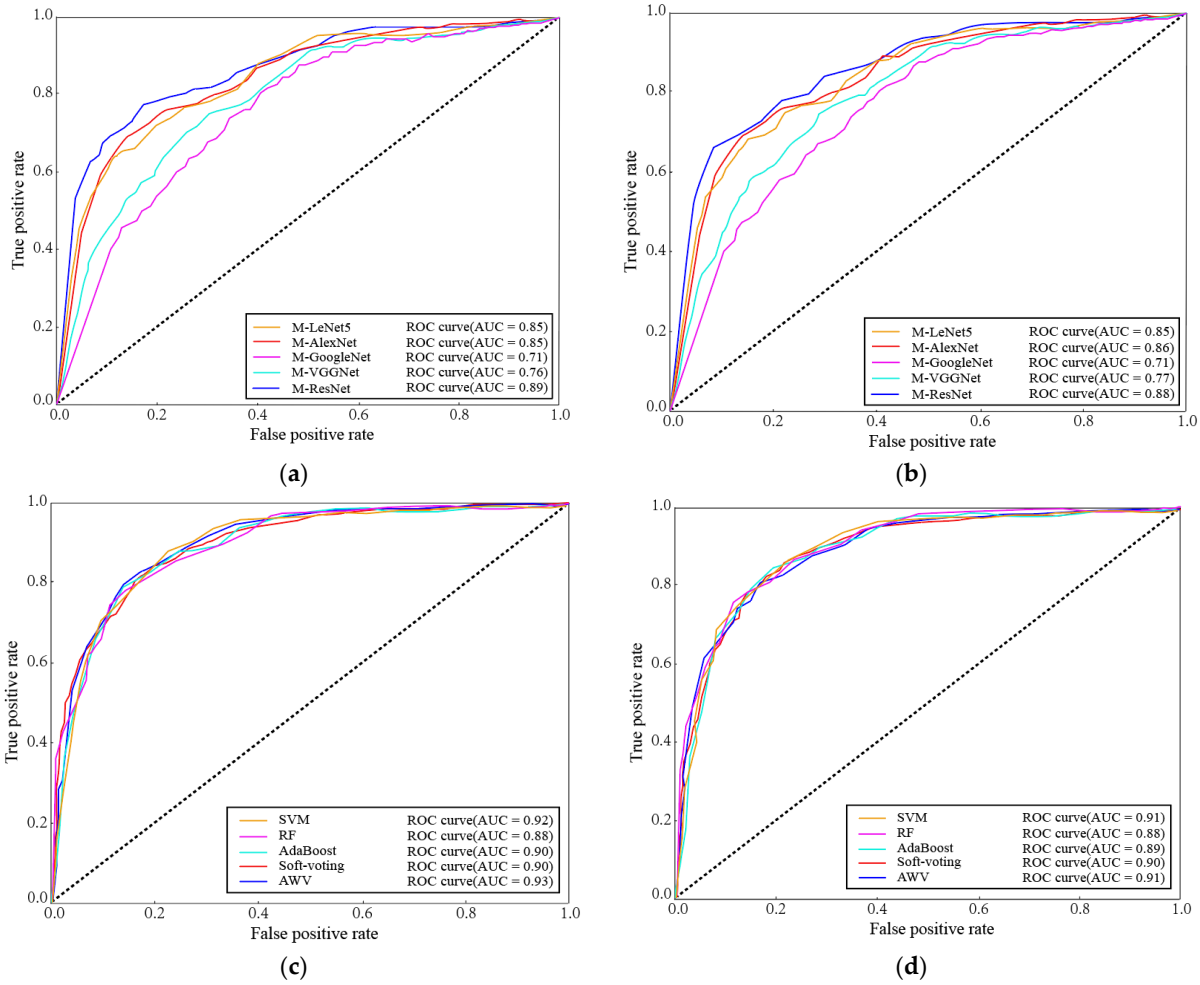

Figure 5. The ROC curves and AUC values: (**a**) M-CNN models on ASL Alphabet, (**b**) M-CNN models on ASLA, (**c**) classifiers on ASL Alphabet, and (**d**) classifiers on ASLA.

To obtain the best model in the feature extraction step, three types of multiple deep CNNs were designed: CNN-3, CNN-4, and CNN-5. CNN-3 uses M-LeNet, M-AlexNet, and M-GoogleNet; CNN-4 adds M-VGGNet to CNN-3; and CNN-5 adds M-ResNet to CNN-4. Figure 6 shows the performance of the ARS-MA models with the three types of CNNs. It shows that increasing the diversity of the higher-depth CNN models (M-GoogleNet, M-VGGNet, and M-ResNet) in the feature extraction step improves recognition accuracy. The accuracies and AUC values of the five classifiers increase as the number of M-CNNs increases, as shown in Figure 6a,b. CNN-5 with the AWV algorithm performed better than other types of CNN models on both datasets.

When $w_{i,j}$ is calculated in Equation (2), two parameters must be assigned. α is related to the relative accuracies among the M-CNN models. Table 8 shows the accuracy of the ARS-MA model by α when λ is 5.

Table 9 shows the accuracy and average test time for an image in ARS-MA as λ varies from 1 to 9 when α is 2. An optimal value of 5 is assigned to λ because it reduces the training time, despite accuracy and average test time being nearly the same over the range of 3 to 7. The proposed ARS-MA model performed best when $\alpha = 2$ and $\lambda = 5$, as shown in Table 7.

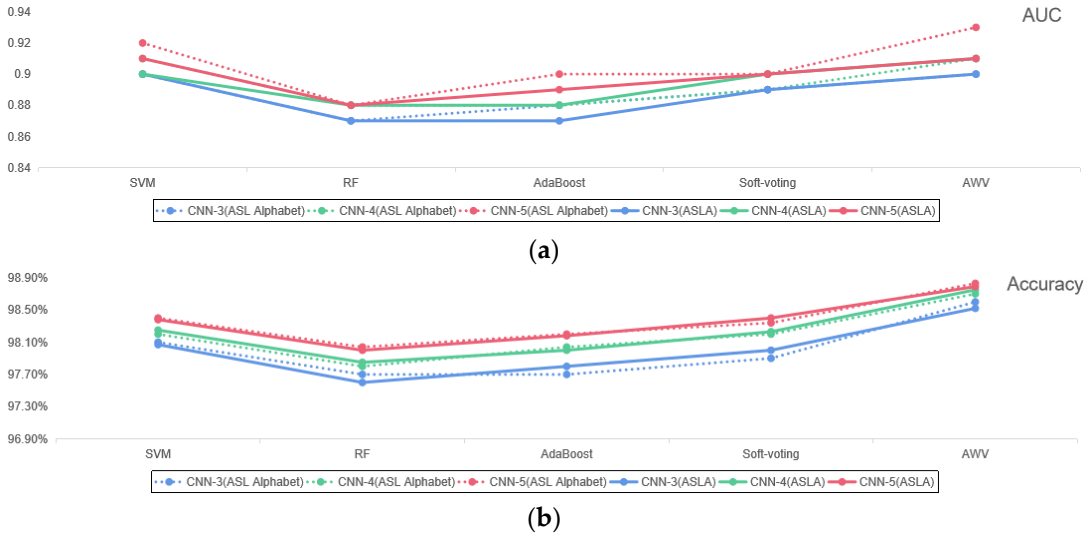

Figure 6. The performance scores for three types of multiple deep CNNs in the ARS-MA model: (**a**) AUC and (**b**) Accuracy.

Table 9. Accuracy and time/image by λ.

λ	ASL Alphabet		ASLA	
	Accuracy	**Average Time/Image**	**Accuracy**	**Average Time/Image**
1	98.8319%	0.312 s	98.7943%	0.324 s
3	98.8327%	0.282 s	98.7945%	0.286 s
5	98.8332%	0.276 s	98.7946%	0.279 s
7	98.7953%	0.273 s	98.7471%	0.271 s
9	98.7139%	0.270 s	98.6816%	0.267 s

Each dataset has 27 non-motion gesture images and two motion gesture images (J and Z) and has different background complexity. Relative to motion and non-motion gestures, the ARS-MA model prioritizes non-motion character with a more significant proportion. ARS-MA has accuracies of 98.98% and 96.88% for non-motion and motion gesture images, respectively. This result means that the accuracy for non-motion images is around 2% higher than for motion images, and that a model for motion image gestures is necessary for better performance.

Table 10 compares the proposed method to other methods for sign language recognition with motions or gestures in recent papers. The proposed ARS-MA model achieved better recognition accuracy on both datasets.

Table 10. Comparison of proposed work and previous works.

Types	**Authors**	**Works**	**Accuracy**
Sign Language Recognition	Marek et al., 2022 [31]	Ensembles for Isolated Sign Language Recognition	73.84%
	Hao et al., 2021 [23]	Self-mutual distillation learning	80%
	Adaloglou et al., 2022 [24]	Inflated 3D ConvNet with BLSTM	89.74%
	Kayo et al., 2020 [18]	Sign Language Translation with STMC-Transformer	96.32%
	Du et al., 2022 [19]	Full transformer network with masking future	96.17%

Table 10. *Cont.*

Types	Authors	Works	Accuracy
ASL recognition	Kania et al., 2018 [26]	Transfer learning using WRN (Wide Residual Networks) on ASL alphabet	93.30%
	Bousbai and Merah, 2019 [27]	Compare custom CNN model and transfer learning using MobileNetV2 on ASLs	97.06%
	Hasan et al., 2020 [13]	Deep learning classification on ASL MNIST dataset	97.62%
	Li et al., 2021 [28]	2D-CNN with joints encoding hand gesture recognition for 14-class ASL alphabet	96.31%
	Kothadiya et al., 2022 [25]	Four different sequences of LSTM and GRU for ASL recognition	95.3%
	Ma et al., 2022 [29]	Two stream fusion CNN for ASL alphabet recognition	97.57%
	Proposed Work (ARS-MA)	The ARS-MA model on 29 classes of ASL Alphabet and ASLA datasets, respectively	98.83% (ASL Alphabet) 98.79% (ASLA)

4. Conclusions

ASL letters are used as an auxiliary language for exceptional cases, such as names, book titles, and letter correction. Building a reliable ASL letter recognition model is essential to improving communication as a tool for hearing-impaired people.

In this paper, the ARS-MA model was proposed which consists of data preprocessing, feature extraction, and classification. Five M-CNN models with different depths and feature capture methods were designed to combine feature extraction and a novel AWV algorithm using an accuracy-based weighted voting proposed to increase accuracy in the final classification step, where two parameters, α and λ, in AWV were introduced to improve the accuracy and consumption time. The best performance was achieved when $\alpha = 2$ and $\lambda = 5$. Three new datasets, ND1, ND2, and ND3, for accuracy-based weighted voting, were created by the results of the multiple deep CNNs. The two datasets used in this paper have two types of images: non-motion gestures and motion gestures (J and Z). The ARS-MA model obtained 98.83% and 98.79% accuracies for the ASL Alphabet and ASLA datasets. In addition, it has accuracies of 98.98% and 96.88% for non-motion and motion gesture images, respectively, and the accuracy for non-motion images is around 2% higher than for motion images.

In the future, the research on increasing the accuracy of motion gesture images will be conducted for a real-time ASL recognition system and it will help hearing-impaired people better communicate.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The generated datasets ND1, ND2, and ND3 during the study can be found in link: https://data.world/ying1211/asl-three-new-datasets.

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

References

1. World Health Organization (WHO). Available online: www.who.int/deafness/world-hearing-day/whd-2018/en (accessed on 1 February 2022).
2. Das, P.; Ahmed, T.; Ali, F. Static Hand Gesture Recognition for American Sign Language using Deep Convolutional Neural Network. *IEEE Sens.* **2020**, *11*, 2.
3. Kamal, S.M.; Chen, Y.; Li, S.; Shi, X.; Zheng, J. Technical approaches to Chinese sign language processing: A review. *IEEE Access* **2019**, *7*, 96926–96935. [CrossRef]
4. National Institute on Deafness and Other Communication Disorders (NIDCD). Available online: https://www.nidcd.nih.gov/health/american-sign-language (accessed on 7 June 2022).
5. Rastgoo, R.; Kiani, K.; Escalera, S. Sign Language Recognition: A Deep Survey. *Expert Syst. Appl.* **2020**, *164*, 113794. [CrossRef]
6. Guo, Y.; Ge, X.; Yu, M.; Yan, G.; Liu, Y. Automatic recognition method for the repeat size of a weave pattern on a woven fabric image. *Text. Res. J.* **2018**, *89*, 2754–2775. [CrossRef]
7. Yu, M.; Guo, Z.-Q.; Yu, Y.; Wang, Y.; Cen, S.-X. Spatiotemporal Feature Descriptor for Micro-Expression Recognition Using Local Cube Binary Pattern. *IEEE Access* **2019**, *7*, 159214–159225. [CrossRef]
8. Kim, J.; Kim, J.; Kim, H.; Shim, M.; Choi, E. CNN-Based Network Intrusion Detection against Denial-of-Service Attacks. *Electronics* **2020**, *9*, 916. [CrossRef]
9. Halder, A.; Tayade, A. Real-time vernacular sign language recognition using mediapipe and machine learning. *ISSN* **2021**, *2582*, 7421.
10. Chuan, C.H.; Regina, E.; Guardino, C. American sign language recognition using leap motion sensor. In Proceedings of the 2014 13th International Conference on Machine Learning and Applications, Detroit, MI, USA, 3–5 December 2014; pp. 541–544.
11. Roy, P.P.; Kumar, P.; Kim, B.G. An efficient sign language recognition (SLR) system using Camshift tracker and hidden Markov model (hmm). *SN Comput. Sci.* **2021**, *2*, 1–15. [CrossRef]
12. Ahmed, W.; Chanda, K.; Mitra, S. Vision based Hand Gesture Recognition using Dynamic Time Warping for Indian Sign Language. In Proceedings of the 2016 international conference on information science (ICIS), Dublin, Ireland, 11–14 December 2017; pp. 120–125.
13. Hasan, M.M.; Srizon, A.Y.; Sayeed, A.; Hasan, M.A.M. Classification of sign language characters by applying a deep convolutional neural network. In Proceedings of the 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT), Dhaka, Bangladesh, 28–29 November 2020; pp. 434–438.
14. Pigou, L.; Dieleman, S.; Kindermans, P.J. Sign language recognition using convolutional neural networks. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 572–578.
15. Jing, L.; Vahdani, E.; Huenerfauth, M.; Tian, Y. Recognizing American sign language manual signs from RGB-D videos. *arXiv* **2019**.
16. Huang, J.; Zhou, W.; Li, H.; Li, W. Sign language recognition using 3d convolutional neural networks. In Proceedings of the 2015 IEEE International Conference on Multimedia and Expo (ICME), Torino, Italy, 29 June–3 July 2015; pp. 1–6.
17. Aloysius, N.; Geetha, M.; Nedungadi, P. Incorporating Relative Position Information in Transformer-Based Sign Language Recognition and Translation. *IEEE Access* **2021**, *9*, 145929–145942. [CrossRef]
18. De Coster, M.; Van Herreweghe, M.; Dambre, J. European Language Resources Association (ELRA). Sign language recognition with transformer networks. In Proceedings of the 12th International Conference on Language Resources and Evaluation, Palais du Pharo, France, 11–16 May 2020; pp. 6018–6024.
19. Du, Y.; Xie, P.; Wang, M.; Hu, X.; Zhao, Z.; Liu, J. Full Transformer Network with Masking Future for Word-Level Sign Language Recognition. *Neurocomputing* **2022**, *500*, 115–123. [CrossRef]
20. Ye, Y.; Tian, Y.; Huenerfauth, M.; Liu, J. Recognizing american sign language gestures from within continuous videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2064–2073.
21. Yu, X.; Zhang, Z.; Wu, L.; Pang, W.; Chen, H.; Yu, Z.; Li, B. Deep Ensemble Learning for Human Action Recognition in Still Images. *Complexity* **2020**, *2020*, 1–23. [CrossRef]
22. Zaidi, S.; Zela, A.; Elsken, T.; Holmes, C.; Hutter, F.; Teh, Y.W. Neural ensemble search for uncertainty estimation and dataset shift. In *Advances in Neural Information Processing Systems*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2021.
23. Hao, A.; Min, Y.; Chen, X. Self-mutual distillation learning for continuous sign language recognition. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 11303–11312.
24. Adaloglou, N.; Chatzis, T. A Comprehensive Study on Deep Learning-based Methods for Sign Language Recognition. *IEEE Trans. Multimed.* **2022**, *24*, 1750–1762. [CrossRef]
25. Kothadiya, D.; Bhatt, C.; Sapariya, K.; Patel, K.; Gil-González, A.-B.; Corchado, J.M. Deepsign: Sign Language Detection and Recognition Using Deep Learning. *Electronics* **2022**, *11*, 1780. [CrossRef]
26. Kania, K.; Markowska-Kaczmar, U. American Sign Language Fingerspelling Recognition Using Wide Residual Networks. In *International Conference on Artificial Intelligence and Soft Computing*; Springer: Cham, Switzerland, 2018; pp. 97–107.
27. Bousbai, K.; Merah, M. A Comparative Study of Hand Gestures Recognition Based on MobileNetV2 and ConvNet Models. In Proceedings of the 2019 6th International Conference on Image and Signal Processing and their Applications (ISPA), Mostaganem, Algeria, 24–25 November 2019; pp. 1–6.
28. Li, Y.; Ma, D.; Yu, Y.; Wei, G.; Zhou, Y. Compact joints encoding for skeleton-based dynamic hand gesture recognition. *Comput. Graph.* **2021**, *97*, 191–199. [CrossRef]

29. Ma, Y.; Xu, T.; Kim, K. Two-Stream Mixed Convolutional Neural Network for American Sign Language Recognition. *Sensors* **2022**, *22*, 5959. [CrossRef]
30. Dong, X.; Yu, Z.; Cao, W.; Shi, Y.; Ma, Q. A survey on ensemble learning. *Frontiers of Computer Science* **2020**, *14*, 241–258. [CrossRef]
31. Hrúz, M.; Gruber, I.; Kanis, J.; Boháček, M.; Hlaváč, M.; Krňoul, Z. One Model is Not Enough: Ensembles for Isolated Sign Language Recognition. *Sensors* **2022**, *22*, 5043. [CrossRef]
32. Zhang, B.; Qi, S.; Monkam, P.; Li, C.; Yang, F.; Yao, Y.-D.; Qian, W. Ensemble Learners of Multiple Deep CNNs for Pulmonary Nodules Classification Using CT Images. *IEEE Access* **2019**, *7*, 110358–110371. [CrossRef]
33. ASL Alphabet Dataset. Available online: https://www.kaggle.com/datasets/grassknoted/asl-alphabet (accessed on 27 February 2021).
34. ASLA Dataset. Available online: https://www.kaggle.com/datasets/debashishsau/aslamerican-sign-language-aplhabet-dataset (accessed on 27 February 2021).
35. Park, K.V.; Oh, K.H.; Jeong, Y.J.; Rhee, J.; Han, M.S.; Han, S.W.; Choi, J. Machine Learning Models for Predicting Hearing Prognosis in Unilateral Idiopathic Sudden Sensorineural Hearing Loss. *Clin. Exp. Otorhinolaryngol.* **2020**, *13*, 148–156. [CrossRef]
36. Karlos, S.; Kostopoulos, G.; Kotsiantis, S. A soft-voting ensemble based co-training scheme using static selection for binary classification problems. *Algorithms* **2020**, *13*, 26. [CrossRef]
37. Yanmei, H.; Bo, W.; Zhaomin, Z. An improved LeNet-5 model for Image Recognition. In *Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 444–448. [CrossRef]
38. Li, S.; Wang, L.; Li, J.; Yao, Y. Image Classification Algorithm Based on Improved AlexNet. *J. Phys. Conf. Ser.* **2021**, *1813*, 012051. [CrossRef]
39. Zhiqi, Yang. Gesture recognition based on improved VGGNET convolutional neural network. In Proceedings of the IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 12–14 June 2020; pp. 1736–1739. [CrossRef]
40. Lee, S.-G.; Sung, Y.; Kim, Y.-G.; Cha, E.-Y. Variations of AlexNet and GoogLeNet to Improve Korean Character Recognition Performance. *J. Inf. Processing Syst.* **2018**, *14*, 205–217.
41. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
42. Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. How does batch normalization help optimization? *Adv. Neural Inf. Processing Syst.* **2018**, *31*, 2483–2493.
43. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 1–74.
44. Cook, J.; Ramadas, V. When to consult precision-recall curves. *Stata J. Promot. Commun. Stat. Stata* **2020**, *20*, 131–148. [CrossRef]