American Sign Language Alphabet Recognition Using Convolutional Neural Networks with Multiview Augmentation and Inference Fusion

Wenjin Tao[a], Ming C. Leu[a], Zhaozheng Yin[b,*]

[a] *Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology, Rolla, MO 65409, USA*
[b] *Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409, USA*

Abstract

American Sign Language (ASL) alphabet recognition by computer vision is a challenging task due to the complexity in ASL signs, high interclass similarities, large intraclass variations, and constant occlusions. This paper describes a method for ASL alphabet recognition using Convolutional Neural Networks (CNN) with multiview augmentation and inference fusion, from depth images captured by Microsoft Kinect. Our approach augments the original data by generating more perspective views, which makes the training more effective and reduces the potential overfitting. During the inference step, our approach comprehends information from multiple views for the final prediction to address the confusing cases caused by orientational variations and partial occlusions. On two public benchmark datasets, our method outperforms the state-of-the-arts.

Keywords: American Sign Language, Convolutional Neural Networks (CNN), Data augmentation, Fusion

1. Introduction

American Sign Language (ASL) is an important communication way to convey information among the deaf community in North America. Although it is

[*] Corresponding author
Email address: yinz@mst.edu (Zhaozheng Yin)

primarily used by people who have hearing or speech difficulties, similar signs also can be used in Natural User Interface (NUI) systems to realize human-computer/robot interaction by hand gestures. Its automatic recognition using various sensing devices has been studied extensively for decades with significant progress having been made. There are mainly two categories of sensing devices used in those studies: (1) wearable devices, such as a cyber glove embedded with a flex sensor or an Inertial Measurement Unit (IMU) sensor, and a set of trackable markers of a motion capturing system; and (2) non-wearable devices, or markerless vision-based devices, such as a RGB camera or a depth camera. Wearable devices directly sense the hand status like adjacent joints' angles, spatial positions and movements, which can provide fairly precise information of the hand [1, 2]. However, they are still too heavy and uncomfortable for daily use. Markerless vision-based recognition has been increasingly popular recently because it does not need sensors attached to a human and the low-cost vision/depth cameras such as Microsoft Kinect are commercially available. However, it is still challenging to recognize ASL signs because of the complexities of these signs, high interclass similarities, large intraclass variations, and constant finger occlusions.

1.1. Related Work

In this paper, we focus on recognizing the alphabet of American Sign Language (ASL). In general, the ASL alphabet recognition task is formulated as two subtasks: feature extraction and subsequent multiclass classification. Researchers have been using different methods to extract discriminative features and create powerful classifiers.

Pugeault and Bowden [3] applied Gabor filters to extract features from both color and depth images at 4 different scales. Then a multiclass random forest classifier was used to recognize the 24 static ASL alphabet signs. They had 49% recognition rate in the leave-one-out experiment. Half of the signs could not be recognized, showing that Gabor filters cannot capture enough discriminative information for differentiating different signs. In addition, they developed a realtime recognition system which provides an interface for the user to select the desired sign among ambiguous ones. It is worth mentioning that they publicized their dataset well and this dataset has been the most common benchmark in this research area, as surveyed in the following.

Wang et al. [4] also used color and depth images for recognition. They proposed a Superpixel Earth Mover's Distance (SP-EMD) metric to measure the distance between two signs based on the shape, texture and depth information. Then a template matching technique was utilized for the sign classification. They reported 75.8% recognition rate on the benchmark dataset. Some researchers only focused on either color or depth image. Maqueda et al. [5] deployed a Volumetric Spatiograms of Local Binary Patterns (VS-LBP) descriptor on color videos or images, without using depth images, for extracting spatio-temporal features. By using a Support Vector Machine (SVM) classifier, they had 83.7% leave-one-out accuracy on the benchmark dataset. Nai et al. [6] extracted features from only depth images on randomly positioned line segments and used a random forest for classification, with 81.1% accuracy reported in their paper.

Some studies attempted to exploit the 3D information embedded in the depth images (3D approach). Kuznetsova et al. [7] implemented an Ensemble of Shape Function (ESF) descriptor [8] on the 3D point cloud for feature extraction and a multi-layered random forest for classification. Zhang et al. [9] proposed a Histogram of 3D Facets (H3DF) descriptor to encode the 3D shape information of different hand gestures. Then they used a SVM with a linear kernel for the classification step and got 73.3% in the leave-one-out accuracy. Later, Zhang and Tian [10] combined their H3DF with a dense sampling method and achieved an improved accuracy of 83.8%. Rioux-Maldague and Giguére [11] created a mask from the depth image and applied it on the intensity image to filter out the hand region to form the intensity features. Six binary images were generated using cross-sections of depth images to form depth features, which were then fed into a Deep Belief Network (DBN) and achieved 77% recall and 79% precision on the benchmark dataset. These 3D approaches are promising to achieve better performance than image representations due to the extra dimension. However, the 3D point cloud obtained from the depth image is sparse at the regions with large gradients and absent at the occluded areas, which affects the overall performance. To fully exploit the 3D benefits from the depth image, some 3D reconstruction methods can be used to recover more valuable information.

Due to the articulated structure of hands, some studies implemented a hand part segmentation step before the gesture recognition (bottom-up approach). Keskin et al. [12] extracted depth comparison features from depth images following the method proposed by Shotton et al. [13] and fed them into a per-pixel random forest classifier. The final predicted label for the whole image is determined by majority voting. They reported their leave-one-out recognition rate as 84.3% on the benchmark dataset. Furthermore, they introduced multi-layered random forests in classifying hand parts to estimate its pose. This classifier is trained using synthetic depth images which have the parts' groundtruth of a hand. To generate more realistic training data for per-pixel hand part classification, a colored latex glove was employed in the research of Dong et al. [14]. They added kinematic constraints on the estimated joint locations to improve the localization accuracy, based on which 13 key angles of the hand skeleton were extracted and fed into a random forest classifier, resulting in 70% recognition rate on the benchmark dataset. One of the major drawbacks for these bottom-up approaches is that the sign recognition performance is highly dependent upon the result of the hand part segmentation, and it is challenging to improve the performance of the hand part segmentation because of the high complexities and constant occlusions.

Recently, deep learning methods such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have demonstrated their extraordinary performance in various classification and recognition tasks. For example, in the traffic sign classification task and the ImageNet challenge, the CNN systems achieved even better performances than those of humans [15, 16]. Unlike the handcrafted feature extractor, which is designed to capture only specific patterns, the deep learning based feature extractor is automatically trained to capture the most discriminative features using the real data. Ameen and Vadera [17] introduced a CNN model with both color and depth inputs in the ASL alphabet recognition task. This model has two convolutional layers for each input to extract features from them. Those two sets of features are concatenated into one before being fed into fully connected layers. They reported the accuracy of 80.34% on the benchmark dataset. RNNs have also been utilized for hand gesture recognition tasks and achieved promising results [18].

1.2. Proposed Method

Depth images contain distance information from the camera plane to the objects in the camera view, where each pixel represents a measured distance. Therefore, it is easier to segment the target object in a depth image than a color image. Thus, this research focuses on recognizing finger spelling signs from depth images as follows:

- Considering the challenges of the ASL alphabet recognition task, we choose CNN as the basic model to build the classifier because of its powerful learning ability that has been shown.
- To fully exploit the 3D information provided by depth images, we develop a novel multiview augmentation strategy. It generates more views from different perspectives, in order to augment the perspective variations that cannot be achieved using traditional image augmentation methods.
- To solve the interclass similarity issues caused by perspective variations and partial occlusions, we first make predictions for all individual views and then fuse information from them for the final prediction.

The remainder of this paper is organized as follows. Our proposed methods of multiview augmentation, CNN model, and inference fusion are detailed in Sections 2, 3 and 4, respectively. The experimental setups and experimental results using the public datasets are described in Sections 5 and 6. Finally, Section 7 gives the conclusions of this research.

2. Multiview Augmentation

To train a valid CNN classifier with good performance, a large amount of labeled data needs to be fed into it. However, it is always time-consuming and costly to collect enough data with annotated labels. Data augmentation is a common method to solve such an issue, which synthesizes additional data derived from original ones.

Traditionally, data augmentation refers to implementing a series of image transformation techniques on the original images, which consist of rotating, scaling, shifting, flipping, shearing, etc. The image transformation is able to introduce more variations and still keep the recognizable features. However, the basic image transformation cannot introduce realistic variations of different perspectives (e.g., out-of-plane transformations in the real world), which are common for hand gestures because they are highly perspective-dependent.

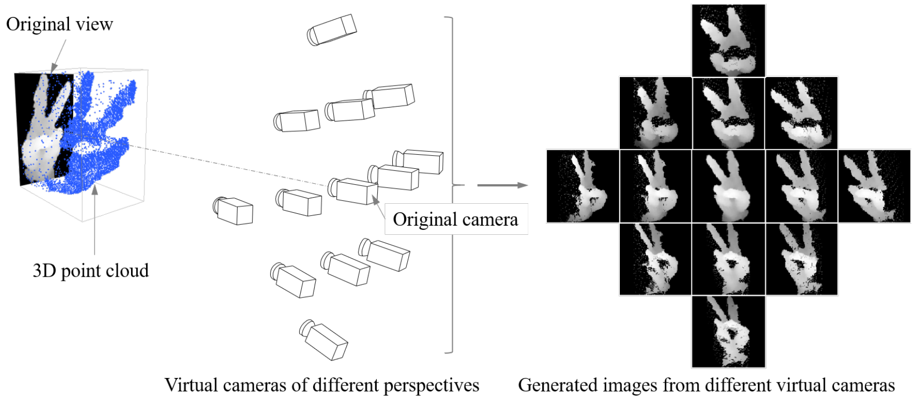

Original view
3D point cloud
Original camera
Virtual cameras of different perspectives
Generated images from different virtual cameras

Figure 1: Multiview augmentation strategy.

To synthesize those perspective variations, we propose a multiview augmentation strategy illustrated in Figure 1. A hand gesture is represented as a depth image in its original view, from which a 3D point cloud is obtained. Then additional virtual cameras are set up and oriented to the point cloud with different perspectives. Finally, a set of additional views are generated from those distributed virtual cameras. The central image on the right hand side in Figure 1 is the original depth image, based on which the other views are generated.

The generation process of a new view is shown in Figure 2. Given a hand depth image I with M pixels (Figure 2(a)), to extract the point cloud $\mathcal{P} = \{p_1, ..., p_m, ..., p_M\}$ from the depth image, each pixel $I(i, j)$ is projected into the 3D space as a point $p_m = (p_m^{(x)}, p_m^{(y)}, p_m^{(z)})$. This projection first translates the origin to the image center and then uses the depth values as the Z values, which is formulated as follows (Figure 2(b)):

$$\begin{aligned} p_m^{(x)} &= j - w/2 \\ p_m^{(y)} &= -i + h/2 \\ p_m^{(z)} &= I(i, j) \end{aligned} \tag{1}$$

where i and j represent the indices of row and column of I, respectively, w and 145 h are the width and height of the depth image, respectively.

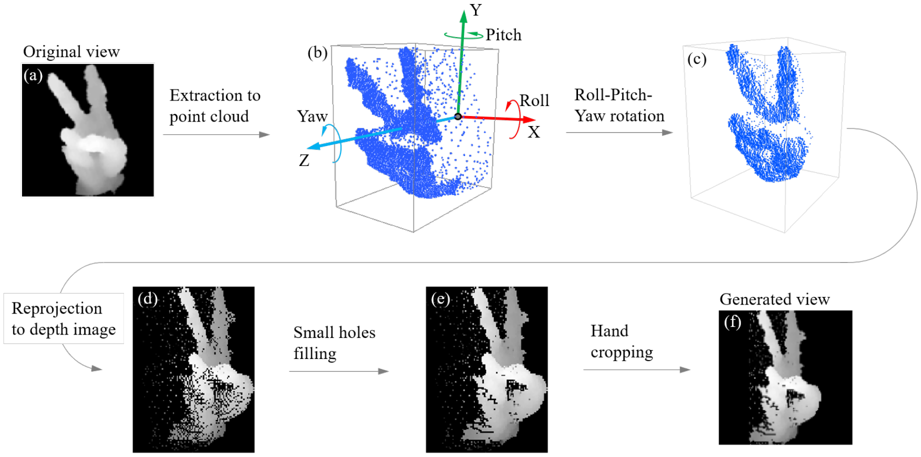


Figure 2: Generation of a new view.

To capture the point cloud from a new perspective, a yaw-pitch-roll rotation transformation on the point cloud around its volume center is implemented. For point p_m in $\mathcal{P}$, its new location after rotation can be calculated by

$$p'_m = R(\alpha, \beta, \gamma)p_m \tag{2}$$

where $R(\alpha,\beta,\gamma)$ is the rotation matrix, α, β and γ represent yaw, pitch and roll angles around z, y and x axes, respectively. It can be further expressed as a multiplication of three orthogonal rotation matrices [19]:

$$
\begin{aligned}
R(\alpha,\beta,\gamma) &= R_z(\alpha)R_y(\beta)R_x(\gamma) \\
&= \begin{bmatrix} \cos\alpha\cos\beta & r_{12} & r_{13} \\ \sin\alpha\cos\beta & r_{22} & r_{23} \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \\
&\mathit{where} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (3) \\
r_{12} &= \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\
r_{13} &= \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\
r_{22} &= \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma \\
r_{23} &= \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma
\end{aligned}
$$

By implementing the yaw-pitch-roll rotation on each point, a new point cloud $\mathcal{P}'$ is generated (Figure 2(c)). Then $\mathcal{P}'$ is reprojected onto a plane to form a new depth image (Figure 2(d)), which is the reverse process of point cloud extraction in Equation 1. After reprojection, the new depth image might have holes because the occluded regions in the original image get exposed in the new one after the yaw-pitch-roll rotation transformation. Those small holes can be filled by interpolation using their neighboring pixels' values (Figure 2(e)). Then the hand region in the new image is cropped and re-centered by removing its surrounding isolated noises (Figure 2(f)).

3. CNN Model

The overall architecture of our CNN model is shown in Figure 3. It is composed of a layered feature extraction module and a classification module.

In the feature extraction module, suppose there are N depth images $X_n, n \in [1, N]$ after data augmentation, they are scaled to the size 32×32 ($width \times height$) and normalized to the interval $[0,1]$, and then fed into three 5×5 convolutional layers for feature extraction. Rectified Linear Unit (ReLU) activation

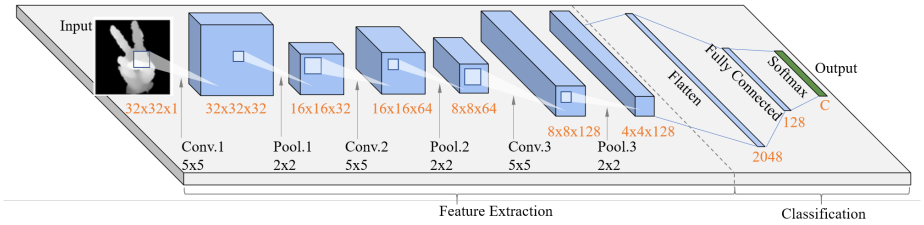


Figure 3: The overall architecture of our CNN model. 'Conv.' and 'Pool.' denote the operations of convolution and pooling, respectively.

function [20] is applied to each convolutional operation. Then each convolutional layer is followed by a 2×2 max pooling layer, which downsamples the previous feature map by a half.

The classification module accepts the $4 \times 4 \times 128$ feature map from the feature extraction module and flattens it as a 2048 feature vector. Then two fully connected layers are used to densify the feature vector to the dimensions of 128 and C sequentially, where C is the number of ASL alphabet sign classes. Then this C-dimensional score vector $S([S_1, ..., S_c, ..., S_C])$ is transformed to output the predicted probabilities with a softmax function as follows:

$$P(y_n = c|X_n) = \frac{\exp(S_c)}{\sum_{c=1}^{C} \exp(S_c)} \tag{4}$$

where $P(y_n = c|X_n)$ is the predicted probability of being class c for sample X_n.

Dropout has been proved to be a powerful regularization technique used to avoid the overfitting, which randomly drops units from the neural network during training [21]. Therefore, it is implemented after each pooling layer in our CNN model.

The process of training a CNN model involves optimization of the network's parameters w to minimize the cost function for the training dataset X. We select the commonly used regularized cross entropy [20] as the cost function, which is

$$\mathcal{L}(w) = \sum_{n=1}^{N} \sum_{c=1}^{C} y_{nc} \log[P(y_n = c|X_n)] + \lambda l_2(w) \tag{5}$$

where y_{nc} is 0 if the ground truth label of X_n is the cth label, and is 1 otherwise. The l_2 regularization term is appended to the loss function for penalizing large weights, and λ is its coefficient.

4. Multiview Inference Fusion

Due to the high interclass similarities, some signs are almost the same from certain perspectives. The inference relying on only one view may not be convincing enough. Therefore, we propose a multiview inference fusion strategy in order to augment the speculation of each individual view, as illustrated in Figure 4.

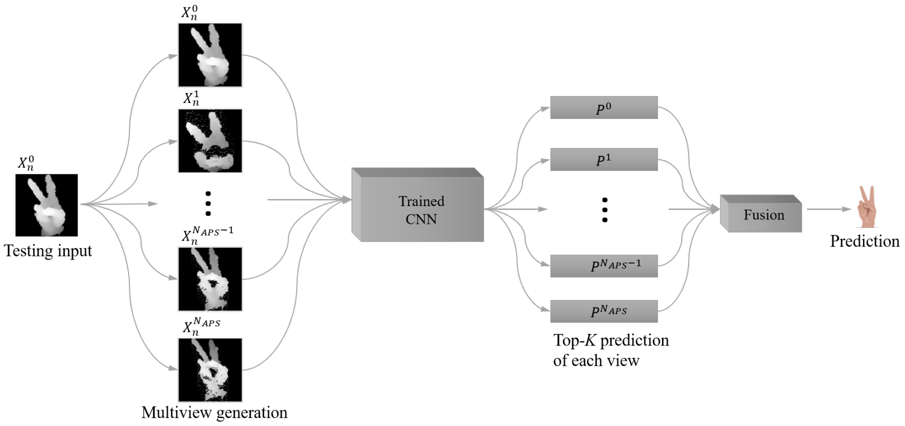


Figure 4: Multiview inference fusion strategy.

In the inference step, suppose the CNN model described in Section 3 has been trained with the augmented dataset, and we have N_{test} depth images for inference. First of all, each query sample is preprocessed to the right input $X_n^0, n \in [1, N_{test}]$ which has the size of 32×32 and the value range of $[0, 1]$. Then, similar to the multiview augmentation process, a set of new views $\{X_n^1, X_n^2, \cdots, X_n^{N_{APS}}\}$ are generated from the original ones X_n^0, where N_{APS} is the number of augmentations per sample. After that, there are $(N_{APS} + 1)$ views $\{X_n^0, X_n^1, \cdots, X_n^v, \cdots, X_n^{N_{APS}}\}$ for the original query sample. Each view

$X_n^v, v \in [0, N_{APS}]$ is inferred individually by the trained CNN to get the probablity distribution P^v of the top-K predicted classes ($K \in [1, C]$, e.g., $K = 5$). Then they are fed into an inference fusion step for the final prediction.

In the inference fusion step, predictions from all individual views are fused together. We introduce the informativity value I^v to evaluate the prediction confidence at each view v. I^v is calculated with Equation 6, which is modified from the Shannon entropy of a discrete probability distribution to vary in the interval of $[0, 1]$.

$$I^v = \frac{\sum_{k=1}^{K} p_k^v \log p_k^v}{\log K} + 1 \tag{6}$$

where v is the index of views and k is the index of top-K candidates. p_k^v represents the probability of the kth class candidate at the vth view. I^v will be close to 0 if all the top-K candidates have similar probabilities (i.e., $p_k^v \approx 1/K$), and 1 if the probability of top-1 class candidate is about reaching 1 (i.e., $p_1^v \approx 1$).

Then every predicted probability p_k^v at the vth view is weighted by I^v of this view. The final predicted label is chosen as the one that maximizes the $I^v p_k^v$ value:

$$\hat{y}_{fusion} = \max_{v} \hat{y}_{fusion}^v \tag{7}$$

where

$$\hat{y}_{fusion}^v = \arg\max_{k} I^v p_k^v \tag{8}$$

5. Experiments

5.1. Datasets

To compare our method with others, we evaluate it on the public ASL alphabet dataset [3]. Some examples of the depth images in this dataset are shown in Figure 5. It has 24 finger spelling signs (‘J’ and ‘Z’ are excluded because they involve finger movement) captured by a Kinect with color and depth images recorded. Those signs were performed by 5 different subjects and each of the

24 signs consists of about 500 to 600 samples. As shown in Figure 5, the hand regions were approximately cropped.

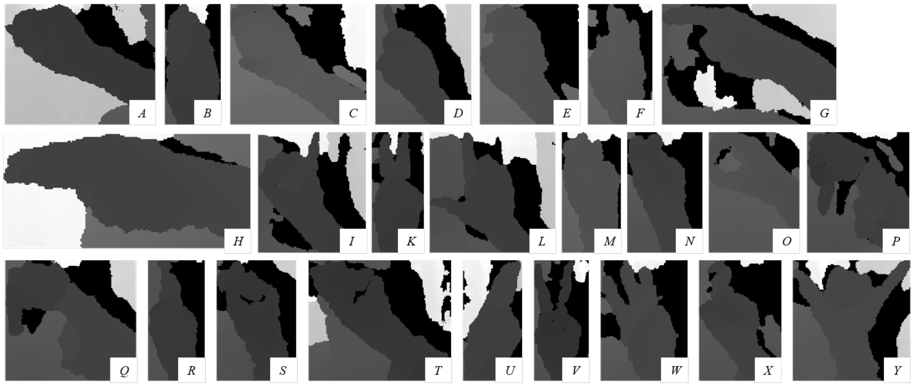


Figure 5: Depth image examples of the 24 signs in the ASL alphabet dataset[3].

To validate the generalization of our method, the NTU digit dataset [22] is also chosen for experiments. This dataset has 10 signs representing digits from 0 to 9 captured by a Kincet containing color and depth images as well. They are performed by 10 different subjects and each sign has 10 samples. Examples of the depth image of the 10 signs in this dataset are shown in Figure 6. Each image contains background and the hand region is not cropped or annotated.

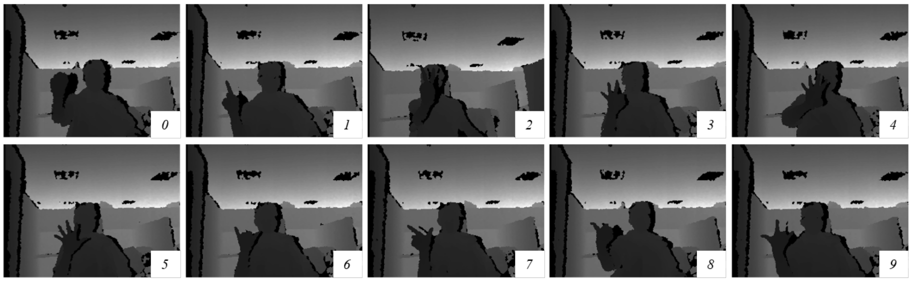


Figure 6: Depth image examples of the 10 signs in the NTU digit dataset [22].

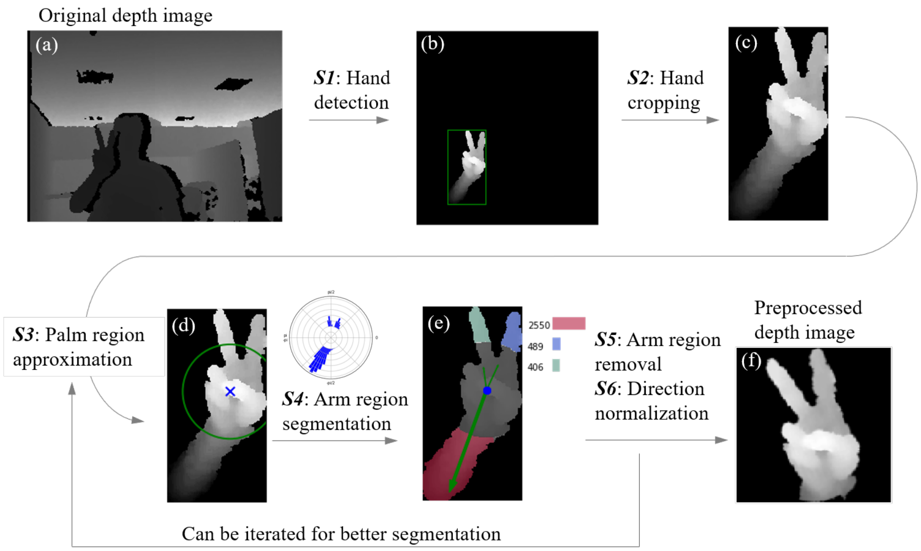


Figure 7: Hand region segmentation. *S1-6* denote the processing steps.

5.2. Preprocessing

The image size of an input sample is a design parameter when building a CNN model and is fixed after the model is created. Thus, it only accepts input samples with the predefined sizes, e.g., our CNN model described in Section 3 needs each input sample to have the uniform size of 32×32 $(width \times height)$.

Samples from the first dataset introduced in Section 5.1 have various sizes (as shown in Figure 5). Although each sample in the second dataset shares the same size (as shown in Figure 6), the size is 640×480 and the hand region is only a small part of the entire image. Therefore, a preprocessing procedure is needed to prepare the data for the CNN model. Taking an image from the NTU digit dataset [22] as an example, this process is illustrated in Figure 7, where *S1-6* denote the processing steps. Suppose there is a raw depth image D (Figure 7(a)) captured by a depth camera and it is assumed that the hand is the closest object to this camera. First, a band-pass filter is applied to filter out the pixels in the range of $[d_{\text{min}}, d_{\text{min}} + \delta]$, where d_{min} is the minimum distance value and δ is the threshold distance that should approximately represent the hand occupation along the direction out of the image. After that, the depth image is reversed using the equation $D' = d_{\min} + \delta - D \quad (if \;\; D \neq 0)$, and then on the new depth image D', hand regions that are nearer to the camera will be brighter, while further regions will be darker. This conversion will let our CNN model focus on the nearer regions which contain more information in distinguishing different signs. There is only one hand and it is the frontmost object in a depth image for both of the datasets. Then the bounding box of the hand is detected (Figure 7(b)) and cropped by using histograms projected onto x and y axes (Figure 7(c)).

The palm center is approximated by calculating the mass center of the depth image and the palm is segmented as a circular region (Figure 7(d)). Then we calculate the polar histogram of the pixels that are outside the palm region, followed by a clustering step. The number of pixels is counted for each cluster. The cluster with the most pixels is segmented as the arm and its direction is taken as the mean of the directions of all its pixels (Figure 7(e)). Finally the arm region is removed, the image is rotated to make the arm direction point down and translated to make the mass center as the image center. Finally, the hand image is reshaped to the size of 32×32 and normalized to the interval of $[0, 1]$ (Figure 7(f)). Note that the processing steps *S3-6* in Figure 7 can be iterated to get a better segmentation result because in some cases the resulted hand still has a large arm area. For example, as shown in Figure 8, the first *S3-6* processing does not remove all the arm region (Figure 8(d)). By implementing the second *S3-6* processing, most of the arm pixels are removed (Figure 8(g)).

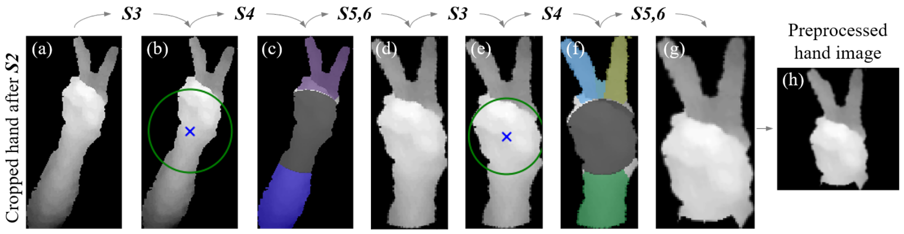


Figure 8: An example of iteration of the processing steps *S3-6* for better segmentation result.

The above preprocessing methods are implemented on the two datasets using tools from the OpenCV library [23], and the resulted samples are shown in Figures 9 and 10, respectively. For the ASL benchmark dataset, the direction normalization step (*S6* in Figure 7) is discarded because some signs (e.g., '*G*' and '*H*') are related to orientations. The implementation of preprocessing and hand segmentation removes the background and prepares the images to have a centered hand on each with a uniform size 32 × 32 for the subsequent CNN training process.

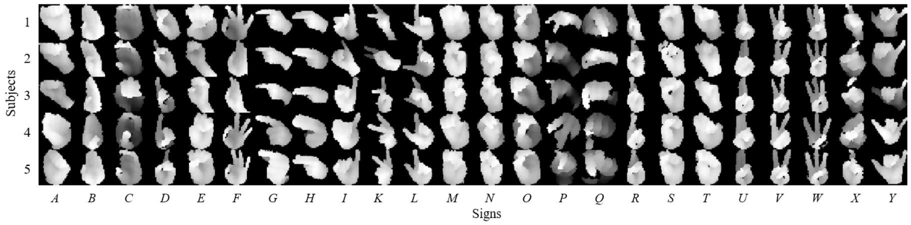


Figure 9: Examples of the 24 signs of each of the five subjects in the preprocessed ASL alphabet dataset.

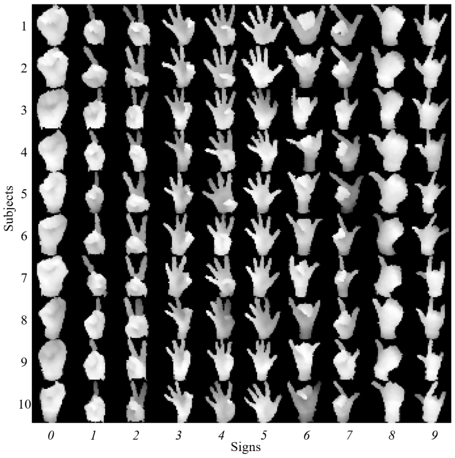


Figure 10: Examples of the 10 signs of each of the ten subjects in the preprocessed NTU digit dataset.

5.3. Evaluation Metric

We conduct comparisons with state-of-the-art recognition results on the above two datasets using the same evaluation policies as in [3], which are half-half and leave-one-out policies. For the half-half policy, one half of the dataset is randomly chosen and fed into the CNN model for training, and the other half is reserved for evaluation. For the leave-one-out policy, the samples from $N_{subjects}-1$ out of $N_{subjects}$ subjects are used for CNN training, and the samples from the left one subject are used for evaluation. We employed a few commonly used metrics to evaluate this multiclass classification performance, which are

- Accuracy

$$Accuracy = \frac{\sum_{n}^{N_{test}} 1(\hat{y}_n = y_n)}{N_{test}} \tag{9}$$

- Precision and Recall

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \end{aligned} \tag{10}$$

- F score

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{11}$$

where $\mathbf{1}()$ in Equation 9 is an indicator function. In Equation 10, True Positive (TP) describes a sample X_n from a certain class y_n that is correctly classified as y_n; False Positive (FP) is defined as a sample X_n from a 'not y_n' class is incorrectly classified as y_n; False Negative (FN) means a sample X_n of the class y_n is misclassified as other 'not y_n' classes. In Equation 11, F score evaluates the overall performance of the of Precision and Recall, which is their harmonic mean in the interval [0,1].

5.4. Some CNN Training Details

TensorFlow [24] is used in creating the CNN model described in Section 3. For the hyperparameters, we set the batch size, learning rate, dropout rate, regularizer as 256, 0.001, 0.1, and 1e-5, respectively. The Adam optimizer [25] is used in training, and the training is stopped after 100 epochs, which takes approximately 2 hours for a leave-one-out experiment on a workstation with one 12 core Intel Xeon processor, 64GB of RAM and one Nvidia Geforce 1080 Ti graphic card.

6. Results and Discussion

6.1. Evaluation of the CNN Architecture

Due to the high architectural complexity and parametric variation of a CNN model, it is not feasible to evaluate all possible architectures and associated parameters (e.g., number of convolutional layers, kernel size, activation function, pooling method, etc.). In this study, a few representative CNN designs with increasing numbers of layers and different parameters are compared to find the optimal design. As shown in Table 1, eight CNN architectures (listed in columns) are selected and their performance of leave-one-out evaluations is compared. We can see that increasing the depth and the number of filters, from the left (arch-i) to the right (arch-viii), improves the evaluation accuracy. Regarding to the convolutional kernel size, the size of 5 outperforms the size of 3. Therefore, the design of arch-viii is chosen as our baseline CNN architecture (see Figure 3).

6.2. Evaluation of the Multiview Augmentation and Inference Fusion Strategies

To evaluate the proposed multiview augmentation and inference fusion strategies, we compare our methods, including MVA (multiview augmentation) and MVA+IF (multiview augmentation and inference fusion) methods, to JA (jittering augmentation) method [15], which has been proved to be an effective method and is commonly used in image classification tasks.

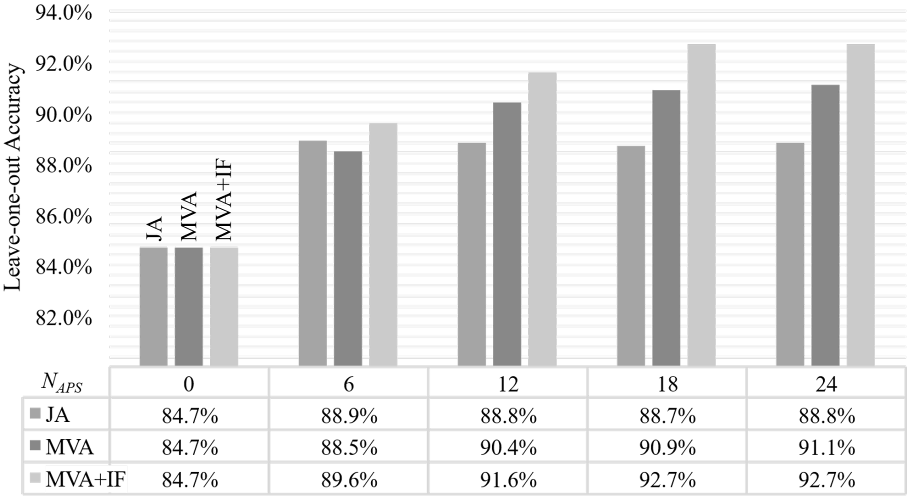
For the MVA and MVA+IF methods, four N_{APS} values 6, 12, 18 and 24 are selected, i.e., new views are generated by implementing yaw-pitch-roll rotation on the extracted point cloud around each axis for $[\pm 10^\circ], [\pm 10^\circ, \pm 20^\circ]$, $[\pm 10^\circ, \pm 20^\circ, \pm 30^\circ]$, and $[\pm 10^\circ, \pm 20^\circ, \pm 30^\circ, \pm 40^\circ]$, yielding 6, 12, 18, and 24

<table>
<tr><td rowspan="12">CNN arch.</td><td>i</td><td>ii</td><td>iii</td><td>iv</td><td>v</td><td>vi</td><td>vii</td><td>viii</td></tr>
<tr><td colspan="8">Input (32 × 32 × 1)</td></tr>
<tr><td>C3-8</td><td>C5-8</td><td>C3-8</td><td>C5-8</td><td>C3-16</td><td>C5-16</td><td>C3-32</td><td>C5-32</td></tr>
<tr><td colspan="8">Maxpool</td></tr>
<tr><td>C3-16</td><td>C5-16</td><td>C3-16</td><td>C5-16</td><td>C3-32</td><td>C5-32</td><td>C3-64</td><td>C5-64</td></tr>
<tr><td colspan="8">Maxpool</td></tr>
<tr><td colspan="2" rowspan="3">FC-1024</td><td>C3-32</td><td>C5-32</td><td>C3-64</td><td>C5-64</td><td>C3-128</td><td>C5-128</td></tr>
<tr><td colspan="6">Maxpool</td></tr>
<tr><td colspan="2">FC-512</td><td colspan="2">FC-1024</td><td colspan="2">FC-2048</td></tr>
<tr><td colspan="8">FC-128</td></tr>
<tr><td colspan="8">FC-24</td></tr>
<tr><td colspan="8">Softmax</td></tr>
<tr><td>Accuracy(%)</td><td>81.8</td><td>82.7</td><td>82.7</td><td>84.2</td><td>83.0</td><td>84.1</td><td>84.7</td><td><b>84.8</b></td></tr>
</table>

Table 1: Comparison of leave-one-out accuracies on the ASL benchmark dataset (without data augmentation) with different CNN architectures (listed in columns). The convolutional layer parameters are denoted as ‘C{kernel size}-{number of output channels}’. The ReLU activation function for each convolutional layer is not shown for brevity. The fully connected layer parameters are denoted as ‘FC-{number of hidden units}’.

augmented views for each sample, respectively. For the MVA+IF method, multiview augmentations are implemented on both the training data and the testing data. Thus, each testing image has multiple vector outputs before the IF step. Then these vector outputs are fused to generate only one probability distribution. While for the MVA method, we only augment the data in the training phase and do not augment the testing data. Therefore, each testing image has only one vector output, from which the final prediction can be made. As for the JA method, the same four N_{APS} values are used; 6, 12, 18, and 24 augmented samples are generated by randomly translating in the range of $[-2, +2]$ pixels, scaling in the range of $[0.9, 1.1]$ ratio, and rotating in the range of [-40, +40] degrees.

The comparisons of leave-one-out accuracies on the ASL benchmark dataset are shown in Figure 11 (the half-half accuracies are not considered for comparison purpose because they are about reaching 100%). All the three augmentation methods have obvious accuracy improvements compared with the model without using data augmentation. For the JA method, using the N_{APS} of 6 improves the accuracy from 84.7% to 88.9%, but continuing to increase N_{APS} from 6 to 24 does not further improve the accuracy. The mean accuracy of the four cases ($N_{APS} = 6, 12, 18, 24$) for MVA is about 88.8%. Although the accuracy of MVA method with the N_{APS} of 6 is a little bit lower than that of JA method, the accuracy increases when increasing the N_{APS}, which outperforms JA after $N_{APS} \geq 12$. The highest accuracy of MVA (91.1%) is from the case of $N_{APS} = 24$, which is 2 percentage points higher than JA.



N_{APS}	0	6	12	18	24
■ JA	84.7%	88.9%	88.8%	88.7%	88.8%
■ MVA	84.7%	88.5%	90.4%	90.9%	91.1%
■ MVA+IF	84.7%	89.6%	91.6%	92.7%	92.7%

Figure 11: Comparison of leave-one-out accuracies on the ASL benchmark dataset using the methods of JA (jittering augmentation), MVA (multiview augmentation) and MVA+IF (multiview augmentation and inference fusion) with different N_{APS} (number of augmentations per sample).

By implementing the multiview inference fusion, more signs are correctly recognized. MVA+IF demonstrates the highest accuracy in all the four cases and for the case of $N_{APS} = 18$ it reaches the best performance of 92.7% accuracy among the three methods. Then increasing the N_{APS} to 24 does not contribute additional improvement.

Overall, the data augmentation techniques, JA and MVA, demonstrate the effectiveness in improving the model performance, because the augmentation process introduces more natural variations to the original dataset to simulate the potential variations in the unseen samples, which pushes the CNN model to learn the most discriminative features and makes the training more robust. Meanwhile, the MVA method outperforms the JA method. It is because the finger spelling signs are highly perspective-dependent, i.e., the appearance of a sign varies significantly from different perspectives, and the MVA method can generate such perspective variations but the JA method can not. Furthermore, the MVA+IF method fuses the predictions of multiple perspectives to make a comprehensive inference, which results in better accuracy than the MVA method.

6.3. Impact of the Number of Top-K Candidates

To find an appropriate number of top-K candidates (the value of K) for the multiview inference fusion step described in Section 4, another set of experiments are conducted on the benchmark dataset. In these experiments, we use different K values, i.e., 3, 5, 7 and 9. Then the leave-one-out evaluation strategy is used and the accuracy evaluated on each of the five subjects is listed in Table 2. We can see that the four 'top-K' cases surpass the 'MVA' case due to the multiview inference fusion step. However, changing the K value from 3 to 9 does not affect the performance much. Therefore, we choose $K = 3$ that can provide enough entries for the fusion process.

Test subject	1	2	3	4	5
MVA	92.74	86.33	94.34	87.73	88.66
MVA+IF, Top-3	93.56	88.51	94.84	91.55	91.69
MVA+IF, Top-5	93.62	88.52	94.78	91.62	91.66
MVA+IF, Top-7	93.64	88.53	94.79	91.61	91.68
MVA+IF, Top-9	93.63	88.53	94.82	91.63	91.71

Table 2: The leave-one-out accuracy (%) tested on each of the five subjects with different numbers of top-K candidates on the ASL benchmark dataset.

6.4. Performance Comparison with State-of-the-Art Methods on the ASL Benchmark Dataset

In this subsection, we compare our results with state-of-the-art performance on the ASL benchmark dataset in terms of accuracy, precision and recall with two evaluation strategies (half-half and leave-one-out). The comparison is summarized in Table 3. The highest accuracies of half-half and leave-one-out strategies in the literature are 100% [9] and 84.3%[12], respectively. For the half-half evaluation, our methods achieve 99.9%, which is almost 100% (there are only about 40 samples misclassified out of 32,831 testing samples). For the leave-one-out evaluation, our CNN model outperforms the state-of-the-art performance even without augmentations. The accuracy is improved by 4% with our implementation of the JA method. By using the MVA method, our model achieves 91% accuracy which is 2% higher than JA. After implementing MVA+IF, the accuracy is improved by another 2 percent. The best accuracy, precision and recall of our results are 92.7%, 93.5% and 92.4%, respectively.

Overall, our CNN model outperforms other methods with the multiview augmentation and inference fusion strategies. It is known that the leave-one-out evaluation is a harder task than the half-half evaluation, because in the half-half experiment, all the testing subjects have already been seen by the CNN model during training; but in the leave-one-out experiment, the testing subject has not been seen. Therefore, the leave-one-out performance can demonstrate how well the trained model could be generalized to a new subject. Our model can reach 93% leave-one-out accuracy, which is a significant improvement compared to the previous best benchmark of 84% and is very promising for practical applications.

6.5. Performance Evaluation on the NTU Digit Dataset

We evaluate the performance of our model on the NTU digit dataset, which also achieves the best accuracies compared to other methods. The comparison is listed in Table 4. The MVA method has 100% and 99.7% for the half-half and leave-one-out accuracies, respectively, which outperforms the results reported in the literatures. The MVA+IF method further improve the leave-one-out

Method	hh-A	hh-P	hh-R	loo-A	loo-P	loo-R
Pugeault et al. (2011) [3]	-	75	53	49	-	-
Keskin et al. (2012) [12]	97.8	-	-	84.3	-	-
Kuznetsova et al. (2013) [7]	87	-	-	57	-	-
Zhang et al. (2013) [9]	98.9	-	-	73.3	-	-
Rioux-Maldague and Giguére (2014) [11]	-	99	99	-	79	77
Dong et al. (2015) [14]	90	-	-	70	-	-
Maqueda et al. (2015) [5]	97.5	-	-	83.7	-	-
Wang et al. (2015) [4]	-	-	-	75.8	-	-
Zhang and Tian (2015) [10]	100	-	-	83.8	-	-
Ma and Huang (2016) [26]	84	-	-	-	-	-
Ameen and Vadera (2017) [17]	-	-	-	80.3	82	80
Nai et al. (2017) [6]	-	-	-	81.1	-	-
Our CNN	99.7	99.7	99.7	84.7	85.8	84.8
Our CNN+JA	99.9	99.9	99.9	88.9	90.2	89.0
Our CNN+MVA	99.9	99.9	99.9	90.9	91.6	90.8
Our CNN+MVA+IF	**99.9**	**99.9**	**99.9**	**92.7**	**93.5**	**92.4**

Table 3: Performance (%) comparison on the ASL benchmark dataset. hh: half-half, loo: leave-one-out, A: accuracy, P: precision, R: recall, '-' denotes that the value is not reported in the paper.

accuracy to 100%, which means that the multiview inference fusion strategy successfully classify the left 0.3% samples that are misclassfied using only MVA.

6.6. Feature Visualization

Although the CNN model demonstrates superior performance on various applications, such as the sign recognition task, it is usually taken as a black box because of its high architectural complexity and tremendous inner parameters, and its hyperparameters are tuned by experience or trial-and-error. To have a better understanding of what the CNN model has learned and what features are extracted by the convolutional operations, we visualize the learned filters and the extracted feature maps of the first convolutional layer since they can be projected into 2 dimensional images, which is shown in Figure 12. The 32 learned filters of the first convolutional layer are presented on the top. It

Method	hh-A	loo-A
Ren et al. (2011) [22]	93.9	-
Zhang and Tian (2015) [10]	97.5	99.0
Our CNN+MVA	100	99.7
Our CNN+MVA+IF	**100**	**100**

Table 4: Performance (%) comparison on the NTU digit dataset. hh: half-half, loo: leave-one-out, A: accuracy, '-' denotes that the value is not reported in the paper.

is difficult to make some intuitive explanations on these 5 × 5 filters, but by reviewing the feature maps obtained from each input using these filters, we can see that some low-level features like edges and curves are extracted. For different signs, the filters are able to identify the discriminative feature elements. For example, the main difference between signs '*M*' and '*N*' is the thumb's position, which is actually learned by the filters (as shown in the feature maps of '*N*', the thumb regions are successfully emphasized compared to the feature maps of '*M*'). Filters and feature maps of the second and third convolutional layers are not presented here because they involve high dimensional information, thus, cannot be projected to images for visualization purpose.

6.7. Failure Case Studies

In this subsection, we discuss the signs that are not correctly classified in the leave-one-out evaluations on the benchmark dataset. The overall mean F scores for the 24 signs are illustrated in Figure 13. The model has great performance ($> 95\%$) on the signs '*B*', '*C*', '*D*', '*F*', '*I*', '*L*', '*O*', '*U*', '*W*', '*X*', and '*Y*'. However, for the signs of '*E*', '*K*', and '*Q*' , the F scores are lower than 85% due to their high subjectwise variations. For example, as shown in Figure 13, different subjects perform the sign of '*K*' in different ways, thus it is difficult for the model to be generalized to the unseen subject in the leave-one-out evaluations.

The confusion matrices and the most confusing sign pairs of the five subjects are shown in Figures 14, 15, 16, 17 and 18, respectively. We can see that,

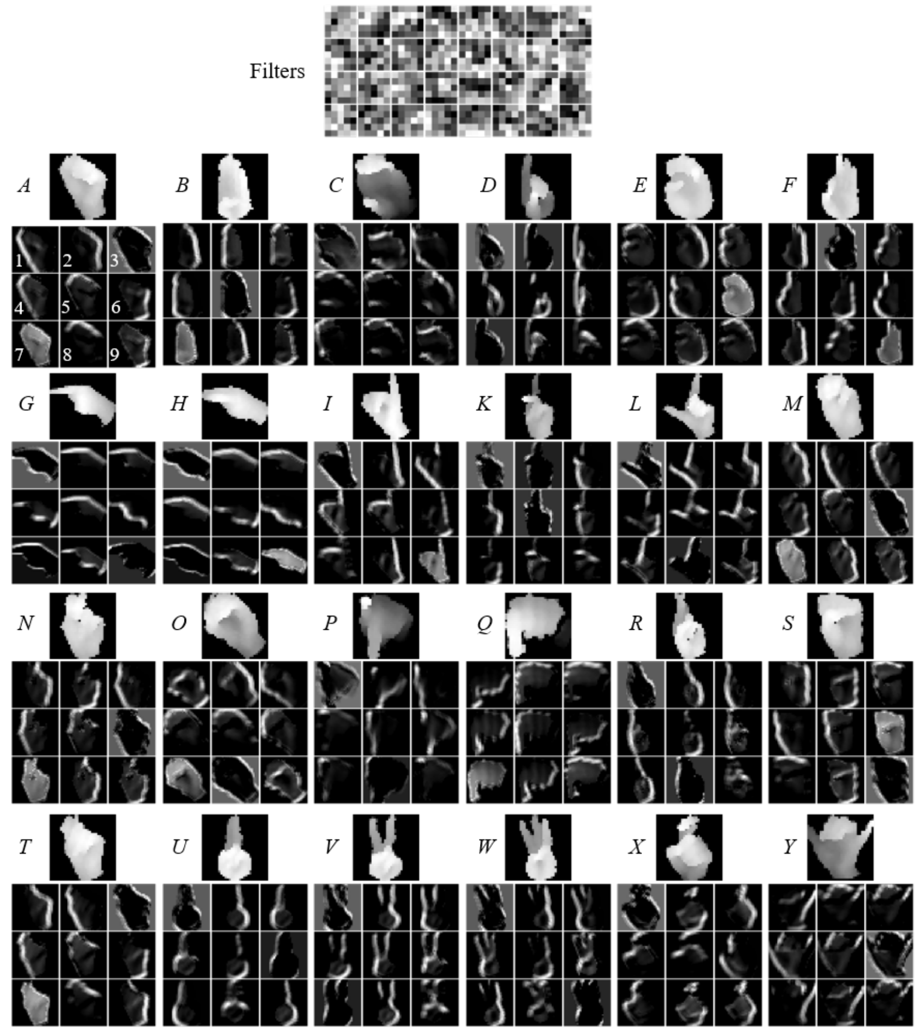


Figure 12: Visualization of the 32 (4 *rows* × 8 *columns*) learned filters (top) of the first convolutional layer, and the top 9 feature maps (the sequence is indexed as shown in *A*'s feature maps) for each of the 24 signs in a trained model on the ASL benchmark dataset. The images with alphabets on the left are the inputs, under which are the feature maps extracted from the inputs using the learned filters.

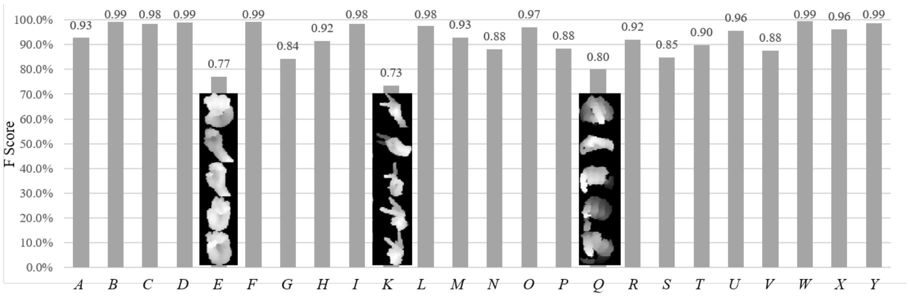


Figure 13: Mean F score of each of the 24 signs in the leave-one-out evaluations on the ASL benchmark dataset.

different subjects show different performance on different signs in the leave-one-out evaluations. For the 1st subject (see Figure 14), there are six confusing pairs severely misclassified, which are '*K*-*G*', '*N*-*T*', '*R*-*K*', '*R*-*U*', '*V*-*K*', and '*X*-*G*'. For example, there are 101 '*V*' misclassified as '*K*' because of the high similarity between them (i.e., both have the index and middle fingers pointing up). For the 2nd subject (see Figure 15), the most confusing pairs are '*G*-*H*', '*V*-*K*', and '*T*-*E*'. For the 3rd subject showing the best performance (see Figure 16), most of the signs are successfully classified except the most confusing pairs '*A*-*T*' and '*K*-*L*', where there are 99 '*A*' and 109 '*K*' misclassified as '*T*' and '*L*', respectively. The most confusing pairs of the 4th (see Figure 17) and 5th (see Figure 18) subjects are '*G*-*H*', '*N*-*M*', '*T*-*N*', and '*E*-*S*', '*V*-*K*', '*Q*-*P*', respectively.

By reviewing these failure cases, we find that the high similarity between the confusing pairs makes it difficult to distinguish them, and the significant subjectwise difference for the same sign makes it difficult to learn this kind of unseen variations beforehand.

To address these failure cases for further improving the performance, some future work can be explored: (1) more subjects can be considered to include more signing styles for training the model; (2) 3D reconstruction can be implemented to recover more information from the depth image than the current 3D

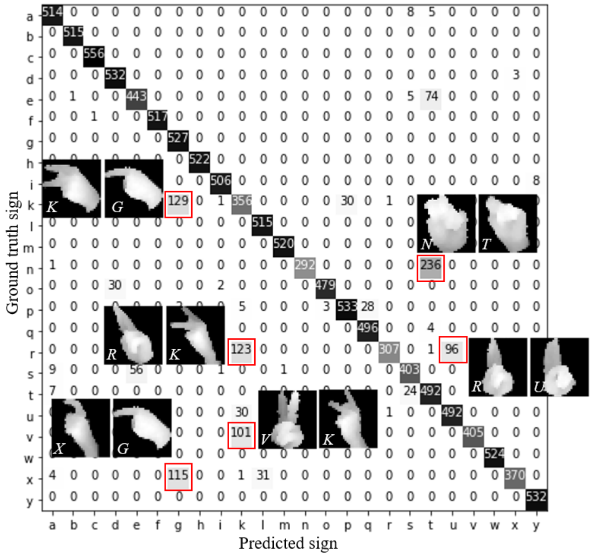


Figure 14: Confusion matrix and the most confusing pairs of the leave-one-out evaluation on the ASL benchmark dataset tested on the 1st subject.

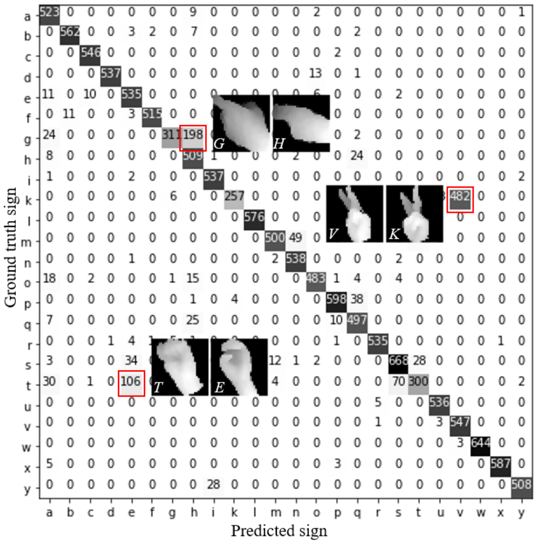


Figure 15: Confusion matrix and the most confusing pairs of the leave-one-out evaluation on the ASL benchmark dataset tested on the 2nd subject.

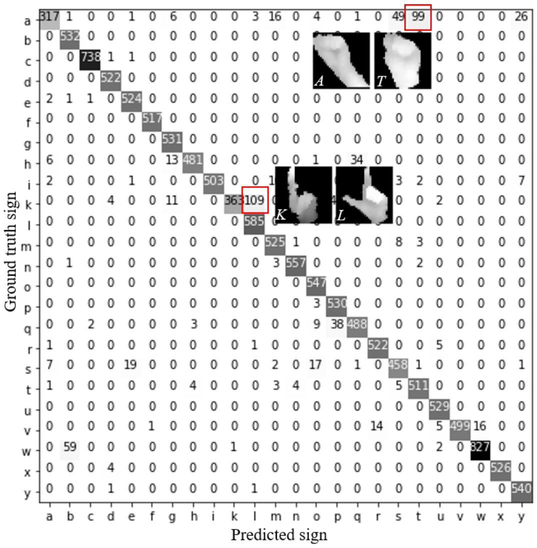


Figure 16: Confusion matrix and the most confusing pairs of the leave-one-out evaluation on the ASL benchmark dataset tested on the 3rd subject.

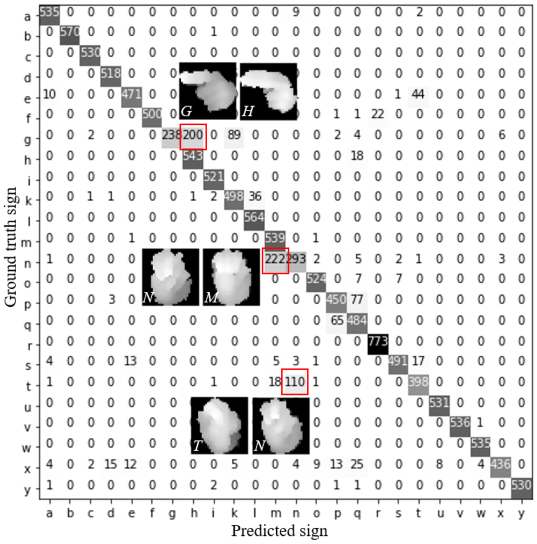


Figure 17: Confusion matrix and the most confusing pairs of the leave-one-out evaluation on the ASL benchmark dataset tested on the 4th subject.

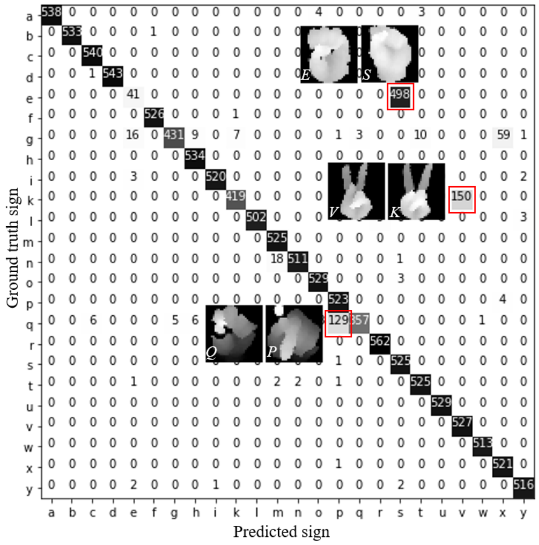


Figure 18: Confusion matrix and the most confusing pairs of the leave-one-out evaluation on the ASL benchmark dataset tested on the 5th subject.

point cloud; (3) the hand skeleton information can be extracted to obtain some skeleton-based features for classification; (4) the RGB images can be included in the model; and (5) the architecture of the CNN model can be explored to improve its performance and efficiency.

7. Conclusions

In this paper, we propose a novel method of multiview augmentation and inference fusion for ASL alphabet recognition from depth images using a Convolutional Neural Network (CNN). Multiview augmentation first retrieves the 3D information embedded in a depth image, and then generates more data for different perspective views. The result has shown that it outperforms the traditional image augmentation methods because it can simulate realistic perspective variations that the traditional methods cannot. Inference fusion copes with the interclass similarity issues caused by perspective variations and finger occlusions. It comprehends information of all individual views, and then outputs the final prediction, which has been proved to be effective in further improving the model's performance. Our method has been successfully evaluated on two public datasets, the ASL benchmark dataset and the NTU digit dataset. The experimental results have demonstrated that our method makes significant improvement compared to the previous work, achieving recognition accuracies of 100% and 93% in the half-half and the leave-one-out experiments, respectively, on the ASL benchmark dataset, and achieving recognition accuracies of 100% for both the half-half and the leave-one-out experiments on the NTU digit dataset.

Acknowledgment

This research work was supported by the National Science Foundation grant CMMI-1646162 on cyber-physical systems and also by the Intelligent Systems Center at Missouri University of Science and Technology. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] C. Oz, M. C. Leu, Recognition of finger spelling of american sign language with artificial neural network using position/orientation sensors and data glove, in: International Symposium on Neural Networks, Springer, 2005, pp. 157–164.

[2] C. Oz, M. C. Leu, Linguistic properties based on american sign language isolated word recognition with artificial neural networks using a sensory glove and motion tracker, Neurocomputing 70 (16) (2007) 2891–2901.

[3] N. Pugeault, R. Bowden, Spelling it out: Real-time asl fingerspelling recognition, in: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, 2011, pp. 1114–1119.

[4] C. Wang, Z. Liu, S.-C. Chan, Superpixel-based hand gesture recognition with kinect depth camera, IEEE transactions on multimedia 17 (1) (2015) 29–39.

[5] A. I. Maqueda, C. R. del Blanco, F. Jaureguizar, N. García, Human–computer interaction based on visual hand-gesture recognition using volumetric spatiograms of local binary patterns, Computer Vision and Image Understanding 141 (2015) 126–137.

[6] W. Nai, Y. Liu, D. Rempel, Y. Wang, Fast hand posture classification using depth features extracted from random line segments, Pattern Recognition 65 (2017) 1–10.

[7] A. Kuznetsova, L. Leal-Taixé, B. Rosenhahn, Real-time sign language recognition using a consumer depth camera, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2013, pp. 83–90.

[8] W. Wohlkinger, M. Vincze, Ensemble of shape functions for 3d object classification, in: Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on, IEEE, 2011, pp. 2987–2992.

[9] C. Zhang, X. Yang, Y. Tian, Histogram of 3d facets: A characteristic descriptor for hand gesture recognition, in: Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on, IEEE, 2013, pp. 1–8.

[10] C. Zhang, Y. Tian, Histogram of 3d facets: A depth descriptor for human action and hand gesture recognition, Computer Vision and Image Understanding 139 (2015) 29–39.

[11] L. Rioux-Maldague, P. Giguere, Sign language fingerspelling classification from depth and color images using a deep belief network, in: Computer and Robot Vision (CRV), 2014 Canadian Conference on, IEEE, 2014, pp. 92–97.

[12] C. Keskin, F. Kıraç, Y. E. Kara, L. Akarun, Hand pose estimation and hand shape classification using multi-layered randomized decision forests, in: European Conference on Computer Vision, Springer, 2012, pp. 852–863.

[13] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, Real-time human pose recognition in parts from single depth images, Communications of the ACM 56 (1) (2013) 116–124.

[14] C. Dong, M. C. Leu, Z. Yin, American sign language alphabet recognition using microsoft kinect, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 44–52.

[15] P. Sermanet, Y. LeCun, Traffic sign recognition with multi-scale convolutional networks, in: Neural Networks (IJCNN), The 2011 International Joint Conference on, IEEE, 2011, pp. 2809–2813.

[16] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[17] S. Ameen, S. Vadera, A convolutional neural network to classify american sign language fingerspelling from depth and colour images, Expert Systems.

[18] D. Avola, M. Bernardi, L. Cinque, G. L. Foresti, C. Massaroni, Exploiting recurrent neural networks and leap motion controller for sign language and semaphoric gesture recognition, arXiv preprint arXiv:1803.10435.

[19] S. M. LaValle, Planning algorithms, Cambridge university press, 2006.

[20] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, `http://www.deeplearningbook.org`.

[21] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., Journal of machine learning research 15 (1) (2014) 1929–1958.

[22] Z. Ren, J. Yuan, Z. Zhang, Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera, in: Proceedings of the 19th ACM international conference on Multimedia, ACM, 2011, pp. 1093–1096.

[23] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools.

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `https://www.tensorflow.org/`

[25] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[26] L. Ma, W. Huang, A static hand gesture recognition method based on the depth information, in: Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2016 8th International Conference on, Vol. 2, IEEE, 2016, pp. 136–139.