Received 26 October 2023, accepted 17 November 2023, date of publication 28 November 2023, date of current version 1 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3336591

METHODS

Two-Dimensional Parallel Spatio-Temporal Pyramid Pooling for Hand Gesture Recognition

FARZANEH JAFARI AND ANUP BASU
Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada
Corresponding author: Farzaneh Jafari (farzane1@ualberta.ca)

This work was supported in part by the Department of Computing Science, University of Alberta, under Natural Sciences and Engineering Research Council (NSERC).

ABSTRACT Hand Gesture Recognition (HGR) plays a crucial role in user-friendly interactions between humans and computers. In recent years, using the Convolutional Neural Network (CNN) has improved the accuracy of image processing problems. Inspired by the high recognition rate of CNN and its efficiency, we propose a model for hand gesture recognition based on CNN and evaluate the results using images with plain and complex backgrounds. Recognizing different hand signs by Two-Dimensional Parallel Spatio-Temporal Pyramid Pooling (2DPSTPP) features with deep learning methods reduces the size of the map, minimizes training complexity, and by paying attention to more details, improves detection performance. The effectiveness of the proposed method is evaluated using regular cross-validation tests on six datasets, namely American Sign Language (ASL), the NUS hand posture dataset I, the NUS hand posture dataset II, the digits dataset, the hand gesture dataset, and the leap gesture recognition dataset.

INDEX TERMS ASL dataset, convolutional neural network (CNN), hand gesture detection, hand gesture dataset, NUS hand posture dataset I, NUS hand posture dataset II, two-dimensional parallel spatio-temporal pyramid pooling (2DPSTPP).

I. INTRODUCTION

Vision and speech are direct interfaces between humans and computers, which are intuitive and user-friendly. In general, people utilize controlling devices, such as remote controls and joysticks, as human-computer interfaces. However, to operate these devices, a trained user or instruction guide is needed. Studies have shown that a Hand Gesture Recognition (HGR) system implemented using data obtained by gloves is more accurate than ones without. However, users feel uncomfortable since their hands are restricted by wearing gloves. Alternatively, detecting hand gestures based on machine vision provides high flexibility without annoying devices, since it allows users to operate the machine by only moving their hands in front of a camera [1]. Several applications that use static and real-time hand gesture recognition systems are gaming [2], smart home interaction control [3], television control [4], controlling the visual environment [5], and visual keywords [6], [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Szidonia Lefkovits.

Data acquisition, hand region segmentation, feature extraction, and hand shape classification according to extracted features, are some steps of hand gesture recognition based on computer vision. Some sensors, like cameras or Kinect depth sensors are used for data acquisition. Some features based on human skin color are used for detecting and localizing gestures. The difficult part of gesture detection is segmentation when the main object is surrounded by a complicated background like the human face and body, or when the colors of background objects are similar to human skin color. Following the above steps, once the semantic information has been extracted from input images, a classifier is used to recognize different gestures [1].

In this study, we propose a general method of hand gesture recognition based on computer vision methods and compare the empirical results of input images with plain and complex backgrounds. Recognizing different hand signs using Two-Dimensional Parallel Spatio-Temporal Pyramid Pooling (2DPSTPP) combined with deep learning reduces the size of the feature maps and minimizes training complexity. Extracting and combining the 2DPSTPP feature vectors help us provide more detailed information into a fully connected layer.

The main contribution of this study is to obtain accurate segmentation of different hand gestures with diverse backgrounds. We present the 2DPSTPP-Net model consisting of four convolutional blocks with four 2DPSTPP pyramid pooling layers extracted from each to improve the accuracy of the hand gesture recognition algorithm. While processing images, the 2DPSTPP-Net model can extract abundant features of hand gestures by combining different dilation rates. Afterward, the pyramid pooling features are concatenated with each other in parallel to obtain dense feature maps. The effectiveness of the proposed method is evaluated using regular cross-validation tests on six datasets, namely American Sign Language (ASL) [8], the NUS hand posture dataset I [9], the NUS hand posture dataset II [10], the digits dataset [11], the hand gesture dataset [12], and the leap gesture recognition dataset [13].

The remainder of this paper is organized as follows. Section II presents some related work on hand gesture detection. Section III describes the framework of the proposed model for deep learning with the pyramid pooling technique. Section IV describes the performance evaluation. Section V provides a brief conclusion.

II. RELATED WORK

Sahoo et al. [1], by considering a pre-trained Convolutional Neural Network (CNN) model, designed an end-to-end fine-tuning method with a score-level fusion technique to recognize hand gestures in a dataset with a low number of gesture images. Three steps were considered for hand gesture recognition: data acquisition, pre-processing, and recognition of hand gestures using the proposed technique. The recognition layer is the most important step of this architecture. It contains a convolutional layer, pooling layer, fully connected layer, and output softmax layer. The kernel slides of the convolutional layer produce the feature map from input images. The pooling layers are used to reduce the feature map dimension by keeping the most important data of the feature map. The fully connected layer is used for classification, and hand gesture classes are predicted by the output softmax layer. Finally, after finishing the network training, the updated weights have constant values.

In [14], a vision-based hand gesture recognition system is designed to detect hand gestures with complicated backgrounds by considering the challenges of detecting skin color and possible dynamic motion of hands. In this study, many problems are considered, such as skin color detection, detection of when a hand reaches the field of the camera view, and full palm detection. Dewi and Juli Christanto [15] used the advantages of Cross-Stage Partial (CSP) networks and Spatial Pyramid Pooling (SPP) layers to improve the performance of the CNN model. CSP is the Yolo-V4 backbone network that is used to improve the learning capacities of the CNN network and reduce the computing bottleneck and memory cost at the same time, and SPP utilizes downsampling in convolutional layers to get the required features and appropriate properties in the max-pooling layers, which are used to construct the model.

Nagi et al. [16] designed a convolution neural network combined with Max-Pooling (MPCNN) for supervised feature learning and classification of hand gestures provided by humans to mobile robots using colored gloves. Color segmentation detected hand contour and morphological image processing was used for smoothing and eliminating noisy edges. This method classifies 6 gesture classes with 96% accuracy. Integrating a CNN with Spatial Pyramid Pooling (SPP) creates a CNN-SPP framework for vision-based hand gesture detection. The spatial pooling layer scales down the size of the feature map to reduce the complexity of training and computation [17]. Practical Deep CNN-based hand gesture recognition with Spatial Pyramid Pooling (SPP) is used to make the model dimension-invariant. The SPP-DCNN model requires 65% fewer parameters and the processing speed is approximately 3 times faster than the classical module. In addition, this method can decode postures from videos with high processing speed [18].

In [19], a two-stage HGR system is proposed to detect hand gestures. First, accurate segmentation extracts the hand from the background. The combination of a dilated residual network, an Atrous Spatial Pyramid Pooling (ASPP) module, and a simplified decoder is used in the segmentation network to determine the hand region in challenging backgrounds. In the next stage, double-channel CNNs are employed to increase recognition accuracy. Experimental results illustrate that the accuracy of the proposed method is approximately 91.71%.

Yang et al. [20] proposed a Pyramid Relation Network (TPRN) based on the Temporal Relation Network (TRN) and Temporal Pyramid Pooling (TPP) to design the temporal relation of video frames effectively. In TRN, relation computation exists in all possible temporal pairs that grow quadratically with video length. However, the TPP layer can accept input with variable length and maintain the temporal order of each bin. TPP obtains the sequence of multiple-scale pyramids, and the TRN is applied on each scale to model the temporal relation of frames at multiple scales. The final prediction is based on all representations of scales. Zhu et al. [21] proposed a model that contains five components, namely input preprocessing, 3D CNN, convolutional Long Short-Term Memory (LSTM), Spatial Pyramid Pooling (SPP), and multimodal fusion. The proposed model can learn short-term spatiotemporal features of hand shapes through a 3D CNN, and then learn long-term spatiotemporal features using convolutional LSTM networks based on the extracted short-term spatiotemporal features. Fine-tuning among multimodal data is used to prevent overfitting when no pre-trained models exist. The results of this model show that it can obtain state-of-the-art recognition accuracy of 51.02% on the validation set of IsoGD and 98.89% on the Shefeld Kinect Gesture (SKIG) dataset.

Jianchun et al. [22] designed a two-stage hand shape recognition architecture in which the first stage is related to the attention mechanism [23] and the Atrous Spatial Pyramid Pooling (ASPP) combination is used to separate gestures from complicated backgrounds. The second stage utilizes a two-stream convolutional neural network that integrates original images with the output of the first stage (the features of the color image and grayscale image) used to improve gesture classification. Wang et al. [24] designed a method with four components: a faster R-CNN, ConvNets, 3D ConvLSTMs, and score fusion. The 3D ConvLSTMs also contain four modules: input preprocessing, 3D convolutional networks, convolutional LSTM, and Spatial Pyramid Pooling (SPP) in which uniform sampling with temporal jitter is used based on the pyramid input to downsample each gesture sequence into a fixed length. Unlike other methods that use a single or cascading multiple types of networks, it uses the heterogeneous network which can allow it to learn spatiotemporal features for hand shape recognition, avoid overemphasizing either spatial or temporal features, and reduce the influence of background by applying it on video sequences.

In [25], a two-stage CNN architecture, called HGR-Net, is proposed for robust recognition of hand gestures against complex backgrounds. The first stage estimates the exact semantic segmentation of hand regions by combining a fully convolutional residual network and Atrous Spatial Pyramid Pooling (ASPP), and the second stage classifies each gesture into the correct classes. The main advantage of this architecture is that it exploits the RGB image and the segmentation map at training time, and recognizes the hand directly from only the 2D RGB frames at testing time. Cui et al. [26] proposed an Improved Atrous Spatial Pyramid Pooling (IASPP) in a convolutional neural network, which creates a feature map by combining cascade and parallel models in Atrous Spatial Pyramid Pooling (ASPP). In addition, IASPP is embedded in an encoder-decoder model (Res-Net) to improve the segmentation performance by integrating details and spatial location information. Zhang et al. [27] designed a hand-raising gesture detection model in the classroom with spatial context augmentation and dilated convolution. This study proposed a Spatial Multi-branch Feature Pyramid Network (SM-FPN) model, focused on the problems with the classical Feature Pyramid Network (FPN) that impede multi-scale features from being fully exploited. Spatial context augmentation was proposed to reduce information loss, and a multi-branch dilated convolution was applied to account for the spatial relationship in the scenes.

Jain et al [28] proposed a sign language recognition system using CNN and Support Vector Machine (SVM). In this method, the optimal filter size is calculated for single and double layers of CNN. Since features are extracted from the dataset and various pre-processing techniques are applied, an SVM with four different kernels and a CNN with single and double layers are used for detection. The experimental results show that the optimal filter size for both single and double-layer CNN is 8×8, with the double-layer CNN achieving an accuracy of 98.58% on the American Sign Language (ASL) dataset. Sharma et al. [29] designed a model based on Gesture-CNN (G-CNN) with a fewer number of model parameters. The features are extracted by sliding a filter window over an input image. The weights of these filters are learned and automatically updated in parallel through extracting features. A non-linear activation function (hyperbolic tangent) is employed to learn the non-linear decision boundaries. The accuracy of 99.96% and 100% is obtained by the G-CNN model for the Indian Sign Language (ISL) and American Sign Language (ASL) datasets, respectively.

Al-Hammadi et al. [30] proposed a novel dynamic system for hand gesture recognition using the integration of multiple deep learning methods. Both local hand shape features and global body configuration features are used to recognize the hand gestures of sign languages. An open-source human estimation posture framework is used to estimate hand region, and a robust face detection algorithm and body part ratio theory are utilized for gesture space estimation and normalization. Also, two 3D CNN instances are used separately for learning fine-grained features of local and global configurations for the classification process.

Miah et al. [31] proposed a three-stage spatial attention-based neural network for hand gesture recognition, in which a feature extractor and a spatial attention module are employed to highlight effective features of the dataset. Extracted features are concatenated with the original images to obtain modality feature embedding. In the same manner, a feature vector and attention map are generated with feature extraction architecture and self-attention technique. After multiplying the attention map and features, the final feature is input into the classification module to evaluate the performance of the model. In [32], a multi-branch attention-based graph and a general deep-learning model are designed to extract all possible types of skeleton-based features in the first step. It also utilizes temporal-spatial features in the second channel using the reverse sequence of the first branch. The general deep learning-based features are extracted using a general deep neural network module and the final feature vector is generated by concatenating the spatial-temporal, temporal-spatial, and general features for the classification section.

Mohammed et al. [33] proposed a Multi-Model Ensemble Gesture Recognition Network (MMEGRN) for skeleton-based effective feature extraction and accurate gesture detection. The architecture of this method consists of four sub-networks, and three spatio-temporal feature classifiers to improve the capabilities of extracting and classifying skeleton sequences. The cyclic annealing learning rate is used to generate a series of models which are combined using the Optimized Weighted Ensemble (OWE) method. In [34], after preprocessing and removing the hand gesture images' backgrounds, it passes through three different streams of feature extraction that

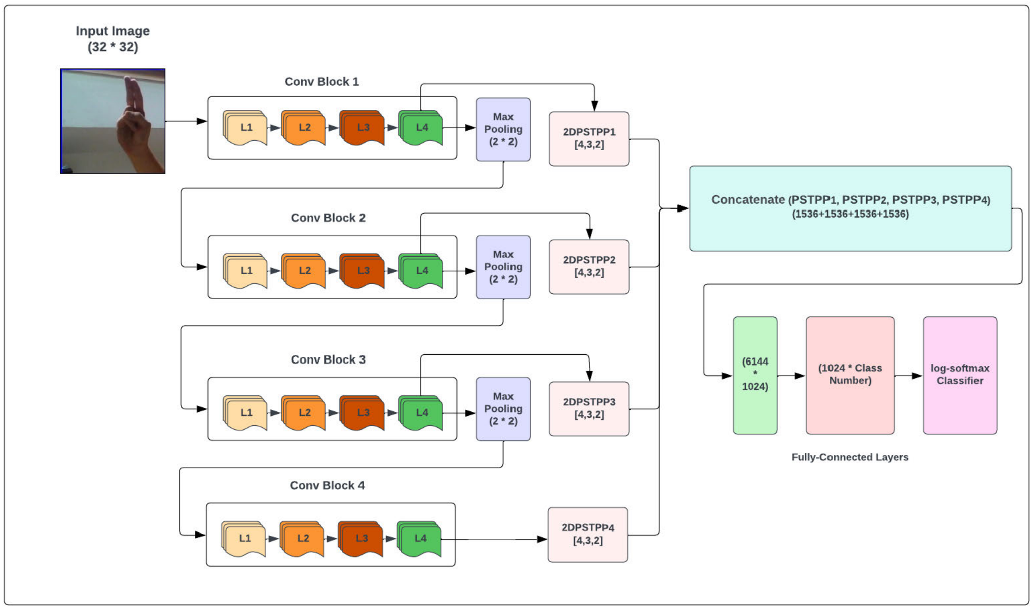

FIGURE 1. The 2DPSTPP-Net framework with four convolutional blocks and four layers in each. The 2DPSTPP is applied to the last feature map of each block.

independently extract their specific features, consisting of three widely used methods named CNN, Gabor filter, and Oriented Fast and Rotated Brief (ORB) feature descriptor. Extracted features from each stream are merged to form the final feature vector.

III. THEORETICAL BACKGROUND AND PROPOSED METHOD

Figure 1 portrays the network construction for the proposed 2DPSTPP-Net method based on the CNN model, in which there are four convolutional blocks, each with four layers. The number of filters for each layer of the first to fourth convolutional blocks are equal to $3 \times 3 \times 64$, $3 \times 3 \times 128$, $3 \times 3 \times 256$, and $3 \times 3 \times 512$, respectively.

2DPSTPP is applied to the last layer of each block. Following this, all pyramid pooling layers (2DPSTPP1, 2DPSTPP2, 2DPSTPP3, and 2DPSTPP4 output vectors) are concatenated with each other, and a single integrated vector is given to a fully connected neural network to classify the hand postures. In other words, after extracting each 2DPSTPP feature from each block, we concatenate them to make a single vector with a fixed length. Extracting the 2DPSTPP features from the last layer of each block extends features forward to the fully connected network, and it also makes the earlier layer learn better since the gradients flow directly into earlier layers. The final 2DPSTPP vector is fed to a fully connected neural network to detect hand postures with high accuracy. In 2DPSTPP-Net, the first fully connected network has 6144 neurons which are directly related to the sum of 2DPSTPPs extracted from all blocks, and the second one has 1024 neurons connected to the output layer with neurons equal to the number of classes. Table 1 demonstrates the summary of the 2DPSTPP-Net network architecture with four convolutional blocks and four 2DPSTPP layers. The number of trainable parameters and forward/backward pass size of the proposed model are equal to 19,011,274 and 17.47 (MB), respectively.

A. TWO-DIMENSIONAL PARALLEL SPATIO-TEMPORAL PYRAMID POOLING (2DPSTPP)

To obtain rich features, we propose a novel pooling layer using a two-dimensional dilation rate as shown in Figure 2. In the first strategy, horizontal and vertical multi-scale feature maps are obtained by parallel mode with different dilation rates $\{2^0, 2^1, \ldots, 2^n\}$. We consider $n = 4$ in our experiments. The outputs of the convolutional layer with the same dilation rates are concatenated like a principal diagonal in a matrix. Table 2 shows the parameters of the 2DPSTPP dilated convolutional module with dilation ratios 1, 2, 4, 8, and 16. The kernel size is 3×3 for all dilation rates except 2^0, where kernel size is equal to 1×1. Thus, the output of the first step of 2DPSTPP can be defined as:

$$y_l = \begin{cases} C_{1,d_i}(x) \oplus C_{1,d_j}(x), \\ \quad i, j = 0. \\ C_{3,d_i}(x) \oplus C_{3,d_j}(x), \\ \quad i = j \,\&\, 1 \leq i, j \leq n. \end{cases} \tag{1}$$

where, $C_{k,d}$ represents the convolutional operation with kernel size k and dilation rate d, d_i and d_j are i-th and j-th dilation rates in the row and column of the matrix respectively; y_l is the output of n+1 patterns in the principle diagonal of the matrix.

To consider all features within the input feature map and deal with the problem of losing features, we utilize

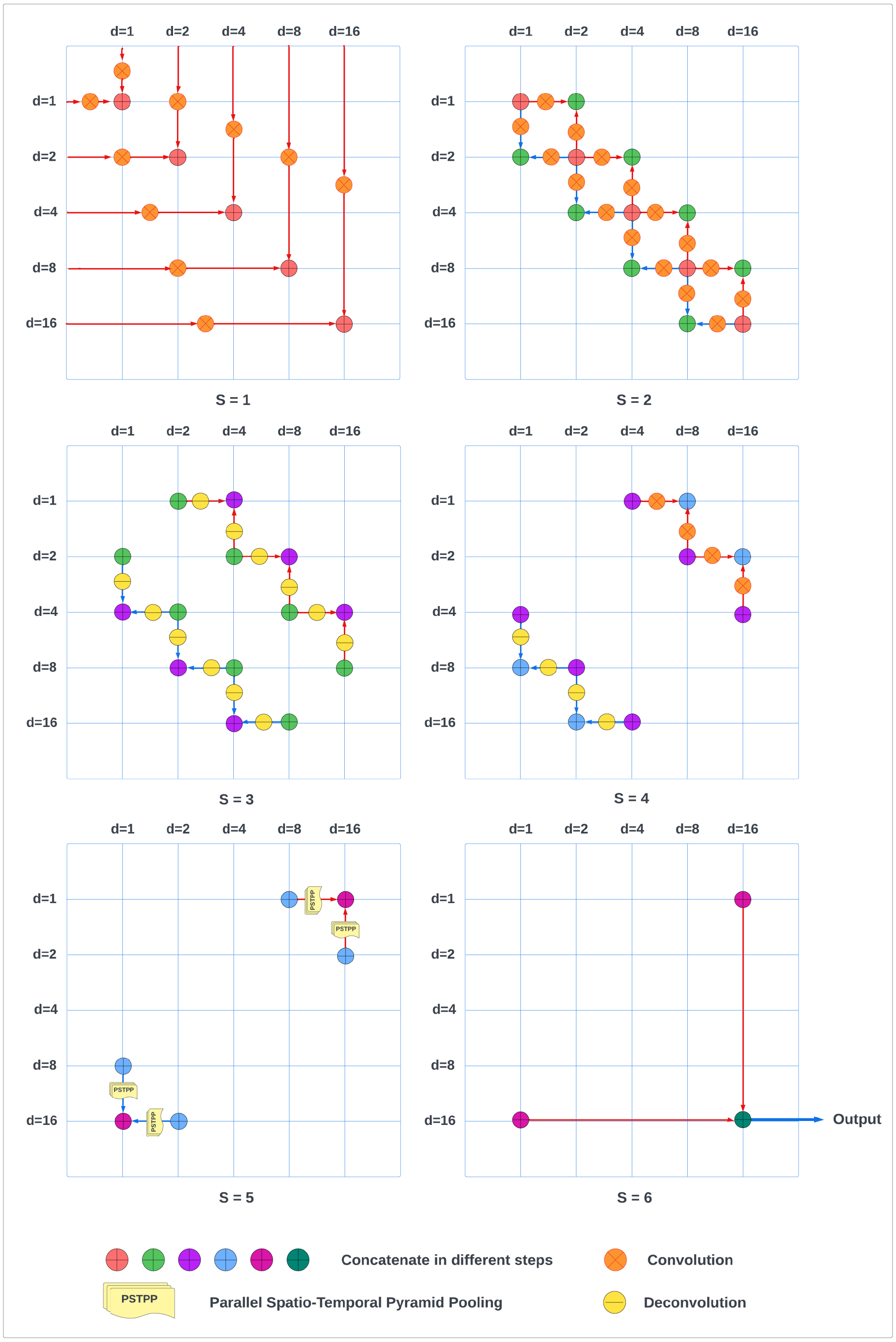

FIGURE 2. **The outline of extracting the 2DPSTPP feature vector in six steps.**

convolutional and deconvolutional layers in different steps. From Step 2 to Step 5, the method contains two diagonals, obtained by the first step, which are placed above and below the principal diagonal. The main purpose of the proposed pyramid pooling is to integrate all of the potential convolutional layers with different dilation rates to utilize all the features obtained during the identification process. The output of the second step of 2DPSTPP is obtained by concatenating the convolution layers of the previous step's feature maps with different dilation rates as defined in Equations 2 and 3:

$$z_{a_l} = \begin{cases} C_{1,d_i}(y_1) \oplus C_{3,d_j}(y_2), \\ \quad i = 0 \;\&\; j = 1. \\ C_{3,d_i}(y_l) \oplus C_{3,d_j}(y_{l+1}), \\ \quad 1 \leq i \leq n-1 \;\&\; j = i+1. \end{cases} \tag{2}$$

$$z_{b_l} = \begin{cases} C_{3,d_i}(y_2) \oplus C_{1,d_j}(y_1), \\ \quad i = 1 \;\&\; j = 0. \\ C_{3,d_i}(y_{l+1}) \oplus C_{3,d_j}(y_l), \\ \quad 2 \leq i \leq n \;\&\; j = i-1. \end{cases} \tag{3}$$

where, z_{a_l} and z_{b_l} denote the output of the above and below diagonal elements respectively, which are obtained from the principle diagonal. The number of feature maps for both diagonals is equal to $2 \times n$.

In the next step, we obtain deconvolutional layers using the second step outputs using Equations 4 and 5:

$$x_{a_l} = \begin{cases} D_{1,d_i}(z_{a_1}) \oplus D_{3,d_j}(z_{a_2}), \\ \quad i = 0 \;\&\; j = 2. \\ D_{3,d_i}(z_{a_l}) \oplus D_{3,d_j}(z_{a_{l+1}}), \\ \quad 1 \leq i \leq n-2 \;\&\; j = i+2. \end{cases} \tag{4}$$

$$x_{b_l} = \begin{cases} D_{3,d_i}(z_{b_2}) \oplus D_{1,d_j}(z_{b_1}), \\ \quad i = 2 \;\&\; j = 0. \\ D_{3,d_i}(z_{b_{l+1}}) \oplus D_{3,d_j}(z_{a_l}), \\ \quad 2 \leq i \leq n \;\&\; j = i-2. \end{cases} \tag{5}$$

where, $D_{k,d}$ defines the deconvolution operation with kernel size k and dilation rate d, x_{a_l} and x_{b_l} represent the third step's output of deconvolutional layers with different dilation rates. The number of output feature maps in this step for each diagonal is equal to $n - 1$. The total number of feature maps obtained by different dilation rates is $2 \times (n - 1)$.

In the fourth step, we concatenate convolutional layers for the top diagonal and deconvolutional layer for the opposite diagonal using Equations 6 and 7:

$$w_{a_l} = \begin{cases} C_{3,d_i}(x_{a_1}) \oplus C_{3,d_j}(x_{a_2}), \\ \quad i = 0 \;\&\; j = 3. \\ C_{3,d_i}(x_{a_l}) \oplus C_{3,d_j}(x_{a_{l+1}}), \\ \quad 1 \leq i \leq n-3 \;\&\; j = i+3. \end{cases} \tag{6}$$

$$w_{b_l} = \begin{cases} D_{3,d_i}(x_{b_2}) \oplus D_{1,d_j}(x_{b_1}), \\ \quad i = 3 \;\&\; j = 0. \\ D_{3,d_i}(x_{b_{l+1}}) \oplus D_{3,d_j}(x_{b_l}), \\ \quad 3 \leq i \leq n \;\&\; j = i-3. \end{cases} \tag{7}$$

where, w_{a_l} and w_{b_l} denote the output feature maps of both equations. The number of feature maps in this step is $2 \times (n - 2)$. Four parallel spatio-temporal pyramid pooling vectors are extracted from each integrated convolutional and

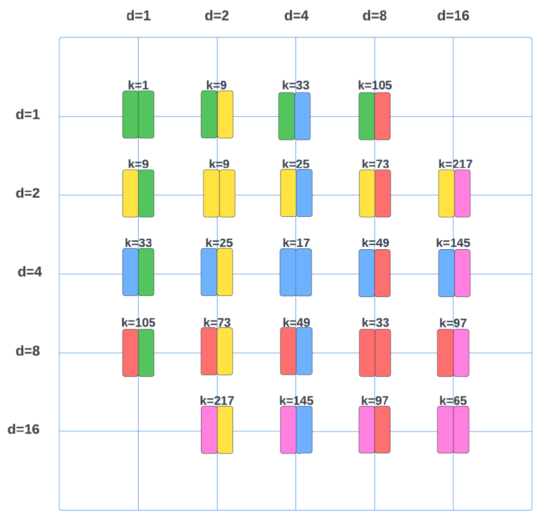

FIGURE 3. The corresponding recursive receptive field (k) of the 2DPSTPP model in different steps.

deconvolutional layers from both diagonal in the previous step by Equations 8 and 9:

$$s_{a_l} = PSTPP(w_{a_1}) \oplus PSTPP(w_{a_2}). \tag{8}$$

$$s_{b_l} = PSTPP(w_{b_1}) \oplus PSTPP(w_{b_2}). \tag{9}$$

where, s_{a_l} and s_{b_l} denote the $2 \times (n - 3)$ output vectors of the above equations. In the last step, all vectors obtained are concatenated using Equation 10:

$$s = s_{a_l} \oplus s_{b_l}. \tag{10}$$

where, s represents four concatenated PSTPP vectors obtained from the above and below diagonals from Step 5. If N_{fm} denotes the total number of feature maps from Step 5 of node (i, j), the total feature length v_l by applying PSTPP on all nodes with $n = 4$ is given by Equation 11:

$$v_l = 4 \times PSTPP \times N_{fm}. \tag{11}$$

The receptive field of the convolution kernel is increased by increasing the dilation rate. For the convolutional layer with kernel size k and dilation rate d, the receptive field (R_f) is obtained by Equation 12 [39]:

$$R_f = (k - 1) \times (d - 1) + k. \tag{12}$$

By stacking two receptive fields, we can access a larger receptive field. If we have two convolutional layers with two different k_1 and k_2 kernel sizes, the concatenated convolutional layer's receptive field is calculated by Equation 13 [39]:

$$k = k_1 + k_2 - 1. \tag{13}$$

Table 3 and Figure 3 illustrate the amount of the receptive field in each step. First, all horizontal and vertical receptive fields are obtained by Equation 12, and diagonals' receptive fields are calculated by Equation 13. The largest receptive field in each diagonal is computed by Equations 14 to

Equations 18, where $R_{f_{k,d}}$ illustrates the receptive filed with kernel size k and dilation rate d.

$$R_{f_{max_{Step_0}}} = max[R_{f_{1,1}}, R_{f_{3,2}}, R_{f_{3,4}}, R_{f_{3,8}}, R_{f_{3,16}}]. \tag{14}$$

$$\begin{aligned} R_{f_{max_{Step_1}}} = max[&R_{f_{1,1}} + R_{f_{1,1}} - 1, \\ &qR_{f_{3,2}} + R_{f_{3,2}} - 1, \\ &R_{f_{3,4}} + R_{f_{3,4}} - 1, \\ &R_{f_{3,8}} + R_{f_{3,8}} - 1, \\ &R_{f_{3,16}} + R_{f_{3,16}} - 1]. \end{aligned} \tag{15}$$

$$\begin{aligned} R_{f_{max_{Step_2}}} = max[&R_{f_{1,1}} + R_{f_{1,1}} \\ &+ R_{f_{3,2}} + R_{f_{3,2}} - 3, \\ &R_{f_{3,2}} + R_{f_{3,2}} \\ &+ R_{f_{3,4}} + R_{f_{3,4}} - 3, \\ &R_{f_{3,4}} + R_{f_{3,4}} \\ &+ R_{f_{3,8}} + R_{f_{3,8}} - 3, \\ &+ R_{f_{3,8}} + R_{f_{3,8}} \\ &R_{f_{3,16}} + R_{f_{3,16}} - 3]. \end{aligned} \tag{16}$$

$$\begin{aligned} R_{f_{max_{Step_3}}} = max[&R_{f_{1,1}} + R_{f_{1,1}} + R_{f_{3,2}} + R_{f_{3,2}} \\ &+ R_{f_{3,2}} + R_{f_{3,2}} + R_{f_{3,4}} + R_{f_{3,4}} - 7, \\ &R_{f_{3,2}} + R_{f_{3,2}} + R_{f_{3,4}} + R_{f_{3,4}} \\ &+ R_{f_{3,4}} + R_{f_{3,4}} + R_{f_{3,8}} + R_{f_{3,8}} - 7, \\ &R_{f_{3,4}} + R_{f_{3,4}} + R_{f_{3,8}} + R_{f_{3,8}} \\ &+ R_{f_{3,8}} + R_{f_{3,8}} + R_{f_{3,16}} + R_{f_{3,16}} - 7]. \end{aligned} \tag{17}$$

$$\begin{aligned} R_{f_{max_{Step_4}}} = max[&R_{f_{1,1}} + R_{f_{1,1}} + R_{f_{3,2}} + R_{f_{3,2}} \\ &+ R_{f_{3,2}} + R_{f_{3,2}} + R_{f_{3,4}} + R_{f_{3,4}} \\ &+ R_{f_{3,2}} + R_{f_{3,2}} + R_{f_{3,4}} + R_{f_{3,4}} \\ &+ R_{f_{3,4}} + R_{f_{3,4}} + R_{f_{3,8}} + R_{f_{3,8}} - 15, \\ &R_{f_{3,2}} + R_{f_{3,2}} + R_{f_{3,4}} + R_{f_{3,4}} \\ &+ R_{f_{3,4}} + R_{f_{3,4}} + R_{f_{3,8}} + R_{f_{3,8}} \\ &+ R_{f_{3,4}} + R_{f_{3,4}} + R_{f_{3,8}} + R_{f_{3,8}} \\ &+ R_{f_{3,8}} + R_{f_{3,8}} + R_{f_{3,16}} + R_{f_{3,16}} - 15]. \end{aligned} \tag{18}$$

B. PARALLEL SPATIO-TEMPORAL PYRAMID POOLING (PSTPP)

Spatial Pyramid Pooling (SPP) solves several limitations of neural networks. The pooling layer is considered after the convolutional layer to reduce the size of the feature map by using max-pooling. This makes it possible to detect information between objects, which is one way that SPP makes network training less complex. SPP can collect more data than other deep learning poolings, allowing the network to classify complex information. Another bonus of SPP is that it can create an output size equal to the input [17], [40]. Temporal Pyramid Pooling (TPP) is similar to the SPP layer in which each feature map from the last convolutional layer is pooled. However, the bins are only segmented horizontally [37].

TABLE 1. Summary of the 2DPSTPP-Net architecture with an input size 32 × 32.

Stage	Layer(type:depth-index)	Output Size	Param
Input	-	[-1, 3, 32, 32]	-
Block 1	Conv2d	[-1, 64, 32, 32]	1,792
	BatchNorm2d	[-1, 64, 32, 32]	128
	ReLU	[-1, 64, 32, 32]	-
	Conv2d	[-1, 64, 32, 32]	36,928
	BatchNorm2d	[-1, 64, 32, 32]	128
	ReLU	[-1, 64, 32, 32]	-
	Conv2d	[-1, 64, 32, 32]	36,928
	BatchNorm2d	[-1, 64, 32, 32]	128
	ReLU	[-1, 64, 32, 32]	-
	Conv2d	[-1, 64, 32, 32]	36,928
	BatchNorm2d	[-1, 64, 32, 32]	128
	ReLU	[-1, 64, 32, 32]	-
	2DPSTPP-1[4,3,2]	[-1, 1536]	-
	MaxPool2d(2, 2)	[-1, 64, 16, 16]	-
Block 2	Conv2d	[-1, 128, 16, 16]	73,856
	BatchNorm2d	[-1, 128, 16, 16]	256
	ReLU	[-1, 128, 16, 16]	-
	Conv2d	[-1, 128, 16, 16]	147,584
	BatchNorm2d	[-1, 128, 16, 16]	256
	ReLU	[-1, 128, 16, 16]	-
	Conv2d	[-1, 128, 16, 16]	147,584
	BatchNorm2d	[-1, 128, 16, 16]	256
	ReLU	[-1, 128, 16, 16]	-
	Conv2d	[-1, 128, 16, 16]	147,584
	BatchNorm2d	[-1, 128, 16, 16]	256
	ReLU	[-1, 128, 16, 16]	-
	2DPSTPP-2[4,3,2]	[-1, 1536]	-
	MaxPool2d(2, 2)	[-1, 128, 8, 8]	-
Block 3	Conv2d	[-1, 256, 8, 8]	295,168
	BatchNorm2d	[-1, 256, 8, 8]	512
	ReLU	[-1, 256, 8, 8]	-
	Conv2d	[-1, 256, 8, 8]	590,080
	BatchNorm2d	[-1, 256, 8, 8]	512
	ReLU	[-1, 256, 8, 8]	-
	Conv2d	[-1, 256, 8, 8]	590,080
	BatchNorm2d	[-1, 256, 8, 8]	512
	ReLU	[-1, 256, 8, 8]	-
	Conv2d	[-1, 256, 8, 8]	147,584
	BatchNorm2d	[-1, 256, 8, 8]	512
	ReLU	[-1, 256, 8, 8]	-
	2DPSTPP-3[4,3,2]	[-1, 1536]	-
	MaxPool2d(2, 2)	[-1, 256, 4, 4]	-
Block 4	Conv2d	[-1, 512, 4, 4]	1,180,160
	BatchNorm2d	[-1, 512, 4, 4]	1,024
	ReLU	[-1, 512, 4, 4]	-
	Conv2d	[-1, 512, 4, 4]	2,359,808
	BatchNorm2d	[-1, 512, 4, 4]	1,024
	ReLU	[-1, 512, 4, 4]	-
	Conv2d	[-1, 512, 4, 4]	2,359,808
	BatchNorm2d	[-1, 512, 4, 4]	1,024
	ReLU	[-1, 512, 4, 4]	-
	Conv2d	[-1, 512, 4, 4]	2,359,808
	BatchNorm2d	[-1, 512, 4, 4]	1,024
	ReLU	[-1, 512, 4, 4]	-
	2DPSTPP-4[4,3,2]	[-1, 1536]	-
	Concatenate(2DPSTPP-1,2,3,4)	[-1, 6144]	-

TABLE 2. Parameters of 2DPSTPP dilated convolution module.

Dilated ratio	2^0	2^1	2^2	2^3	2^4
Receptive field	1×1	5×5	9×9	17×17	33×33
Kernel size	1×1	3×3	3×3	3×3	3×3
Padding	0	2	4	8	16
Stride	1	1	1	1	1

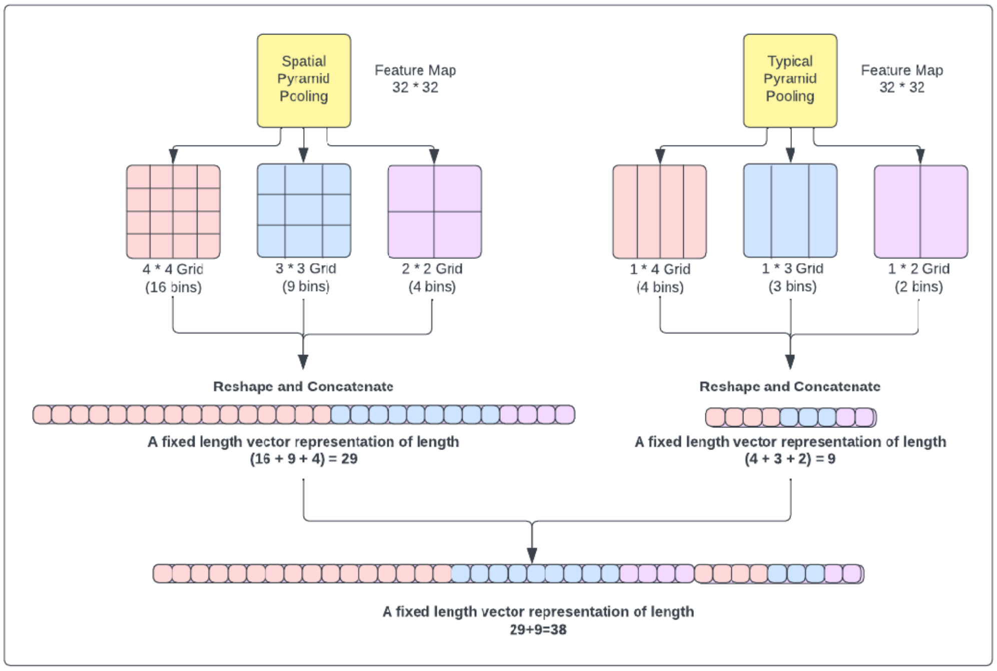

FIGURE 4. PSTPP with [4,3,2] grid size. The result is a vector of 38 features given from a feature map.

TABLE 3. The receptive field of the 2DPSTPP model in different steps. PD = Principle diagonal, APD = Above PD, BPD = Below PD, H = Horizontal, V = Vertical.

Step 0	H	1	5	9	17	33
	V	1	5	9	17	33
Step 1	PD	1	9	17	33	65
Step 2	APD	9	25	49	97	
	BPD	9	25	49	97	
Step 3	APD	33	73	145		
	BPD	33	73	145		
Step 4	APD	105	217			
	BPD	105	217			

Figure 4 shows a PSTPP framework with three grid sizes: four, three, and two. The final result of SPP contains 29 values that have been extracted from the input feature map and connected as a single vector. The three-level TPP has the same layers as SPP. In each step of pyramid pooling, the feature map is divided into fixed portions, and the maximum value of each bin is extracted and appended to a single vector. After extracting SPP and TPP vectors with 29 and 9 lengths respectively for each image, the PSTPP vector with 38 fixed lengths is made by concatenating two SPP and TPP vectors. The integration of SPP and TPP not only preserves the advantages of the SPP method but also improves the results for classification by adding some additional information to the vector.

The aim of using the PSTPP vector is to alleviate the performance of the network by reducing its complexity. Max-pooling extracts the most prominent values from each bin. However, ave-pooling restores the average value of all pixels that lost information necessary to recognize the relationship between different parts of objects. The PSTPP pooling improves the performance of classification by using multiple layers of pooling and extracting different data from images with different bin sizes in each layer.

If the grid size of SPP is equal to $G \times G$, and h and w represent the features map's height and width respectively, the filter sizes $f = f_h = f_w$ is calculated by $f_h = \lceil h/G \rceil$ and $f_w = \lceil w/G \rceil$. As well, the stride height and stride width for pooling are denoted by S_h and S_w, where $S_h = \lceil h/G \rceil$ and $S_w = \lceil w/G \rceil$. The feature map's height and width paddings are calculated by $pad_h = G \times S_h - h$ and $pad_w = G \times S_w - w$ [17]. In the TPP the grid size is equal to $1 \times G$. Thus, filter, stride, and padding are only calculated for the width.

The feature map in the PSTPP vector has a fixed length. v represents the $[1, \ldots, V]$ levels of PSTPP features. The vector length for V-level PSTPP is obtained by Equation 19:

$$PSTPP = w_0 \Sigma_{v=1}^{V} G_v^2 + w_1 \Sigma_{v=1}^{V} G_v, \tag{19}$$

where, $G_v > 0$ is the index of PSTPP grid size (SPP=$G_v \times G_v$ and TPP=$1 \times G_v$), and w_0 and w_1 represent the weightings which are applied to each pooling.

IV. EXPERIMENTS

In this section, experiments are designed to evaluate the performance of the deep learning model with the Two-Dimensional Parallel Spatio-Temporal Pyramid Pooling (2DPSTPP) technique.

A. DATASETS

Six different types of datasets with plain and complex backgrounds, like the American sign language (ASL) [8], the NUS Hand Posture dataset I [9], the NUS Hand Posture

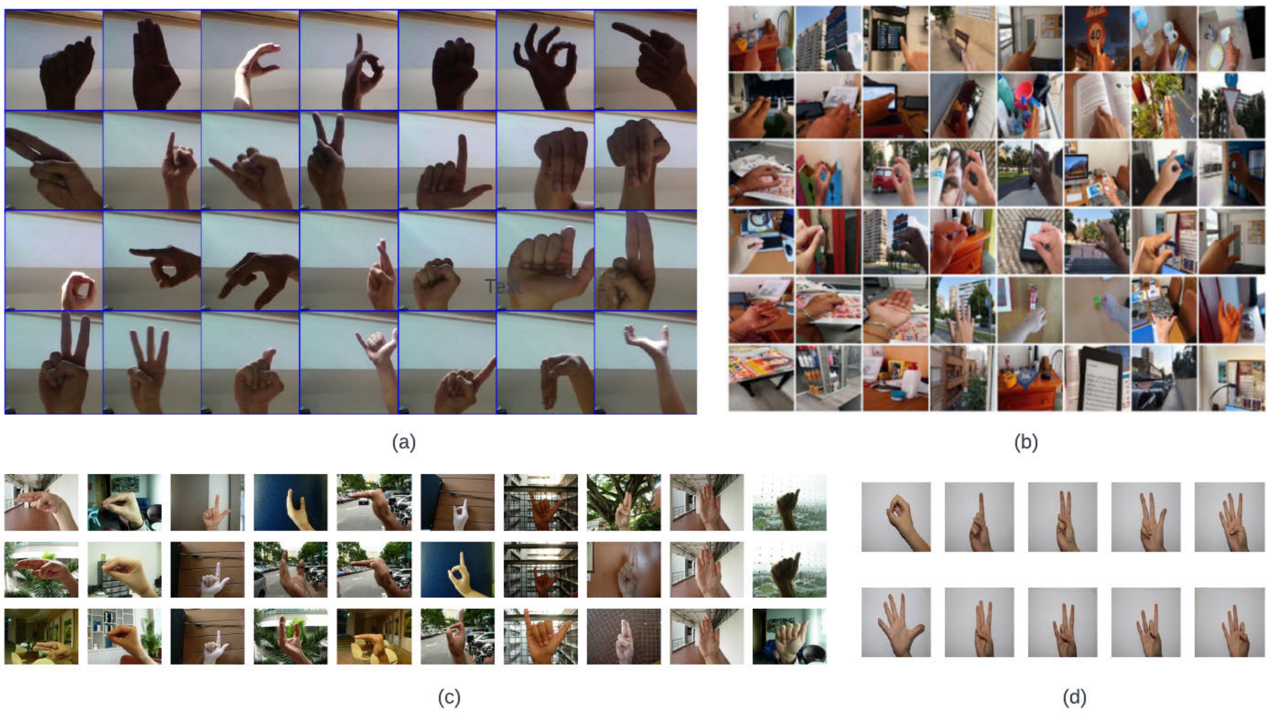

FIGURE 5. Sample images of different hand gesture datasets: (a) the American sign language (ASL) dataset with the plain background, (b) the hand gesture dataset, (c) the NUS dataset with a complex background, and (d) the digits dataset.

dataset II [10], the digits dataset [11], the hand gesture dataset [12], and hand gesture recognition database [13] are used. The ASL, NUS-I, NUS-II, digits, HGD, and leapGestRecog datasets contain 87000, 240, 2000, 2062, 12064, and 20000 images of diverse hand gestures with different backgrounds, respectively. ASL contains 29 posture classes: A to Z alphabets, del, nothing, and space. The NUS hand posture datasets I and II (160×120 pixels) contain only A to J alphabets (10 classes) captured by different hand sizes and scenes. The digits dataset also has 10 hand postures (digits 0 to 9) with a light background. The hand gesture dataset (HGD) contains 6 diverse groups: drag, loupe, none, other, point, and scale are captured under different and complex backgrounds making the dataset more challenging. The hand gesture recognition database is composed of 10 different classes which were performed by 10 different subjects (5 men and 5 women).

B. IMPLEMENTATION DETAILS

We use Python 3.7.12 with CUDA version 11.8.89 for all our experiments. The experiments have been carried out using PyTorch [38], an open-source and optimized tensor library for deep learning [9]. The models are trained at each stage with batch size 32, learning rate 0.0002, and dropout rate 0.5. The models have been experimented with cross-entropy loss function, Stochastic Gradient Descent (SGD) optimizer, and trained with NVIDIA GeForce RTX 2080 SUPER. Based on the GPU limitation, we resized images to 32 $\times$ 32. In this experiment, we consider 80% of total data for training, 10% for validation, and 10% for the test, which are randomly selected from the whole dataset. In addition, we perform 5-fold cross-validation for each dataset.

C. EXPERIMENTAL ANALYSIS

We have four convolutional blocks with four layers in which four 2DPSTPP feature vectors are extracted from the last layer of each block. Table 4 illustrates the performance of the proposed model with four 2DPSTPP layers. In the second row, we only extract Spatial Pyramid Pooling (SPP) instead of Parallel Spatio-Temporal Pyramid Pooling (PSTPP) from each block, and in the third row, we evaluate the performance of the model without 2DPSTPP layers. As is evident from the table, the performance of the method in both training and testing experiments with four 2DPSTPP layers is much better than the other methods.

Figure 6 illustrates the bar charts to compare the test results of our model with 2DSPP and no-2DPSTPP segmentation methods on different datasets. Judging by the results in Figure 6, the effectiveness of the proposed model with four 2DPSTPP layers is demonstrated to be superior. Our method achieves better recognition accuracy on all six datasets that are tested. The test accuracy results for the three datasets (NUS-I, digits, and Leap) are the same at 100%, and the ASL results of 99.98% are near to other superior results. Likewise, the test results of NUS-II and HGD datasets with 99.61% and 98.01% respectively are higher than the other methods.

If we have n convolutional blocks, the number of extracted 2DPSTPP feature vectors is equal to n. Therefore, we can

TABLE 4. The performance of the 2DPSTPP-Net model (test and train accuracy (%)) considering four pooling layers with parallel spatio-temporal pyramid Pooling (PSTPP), spatial pyramid pooling (SPP), and without pyramid pooling layer using all datasets.

Methods	Datasets	Train accuracy (%)	Test accuracy (%)
2DPSTPP	ASL	**99.98**	**99.98**
	NUS-I	**100**	**100**
	NUS-II	**100**	**99.61**
	Digits	**100**	**100**
	HGD-Real	**98.09**	**98.01**
	LeapGestRecog	**100**	**100**
2DSPP	ASL	99.95	99.89
	NUS-I	91.67	95.83
	NUS-II	99.22	98.43
	Digits	91.63	**100**
	HGD-Real	96.78	97.92
	LeapGestRecog	99.90	**100**
With no 2DPSTPP	ASL	98.48	98.40
	NUS-I	95.83	95.83
	NUS-II	98.44	96.85
	Digits	99.61	99.21
	HGD-Real	96.38	96.02
	LeapGestRecog	99.71	99.80

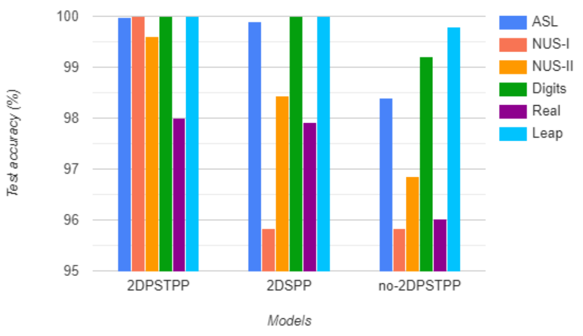

FIGURE 6. Histogram of results on six datasets test sets using different pyramid pooling layers.

achieve 2^n subsets of a set with n elements ($\Sigma_{k=0}^{n}\binom{n}{k} = 2^n$). In this study, we consider $n = 4$, so the subsets of 2DPSTPP layers which can be extracted from different blocks are $\{\{\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\}$. Thus, the total number of subsets is 16. By extracting different pooling layers from different blocks, we can evaluate the effectiveness of the proposed model by considering the different subsets of 2DPSTPP layers extracted from different blocks. Since the number of 2DPSTPP layers affects the feature extraction as well as the segmentation accuracy of networks, we compare the effect of four 2DPSTPP layers on the segmentation accuracy with the other subsets mentioned. The purpose of comparing our model with different layers is to determine which subset has better results during detection. In other words, to increase the detection accuracy, we need to determine the optimum number of layers for our model. Table 5 and Figure 7 compare the performance of the proposed model with three different 2DPSTPP layers. It can be seen that the test results of NUS-II, digits, and Leap datasets with 100% meet the highest accuracy in subset

TABLE 5. The performance of the 2DPSTPP-Net model (test and train accuracy (%)) considering three pooling layers extracted from different convolutional blocks using all datasets.

Methods	Datasets	Train accuracy (%)	Test accuracy (%)
$2DPSTPP_{1,2,3}$	ASL	99.53	99.55
	NUS-I	**100**	87.50
	NUS-II	98.05	99.61
	Digits	98.44	**100**
	HGD-Real	96.04	**97.59**
	LeapGestRecog	**100**	99.02
$2DPSTPP_{1,2,4}$	ASL	99.47	99.25
	NUS-I	**100**	83.34
	NUS-II	98.05	99.22
	Digits	97.43	**100**
	HGD-Real	**97.73**	97.11
	LeapGestRecog	**100**	99.90
$2DPSTPP_{1,3,4}$	ASL	**99.71**	**99.70**
	NUS-I	**100**	**91.66**
	NUS-II	**98.44**	98.43
	Digits	99.61	99.61
	HGD-Real	96.60	97.42
	LeapGestRecog	**100**	**100**
$2DPSTPP_{2,3,4}$	ASL	98.02	98.32
	NUS-I	**100**	87.50
	NUS-II	97.66	**100**
	Digits	97.82	**100**
	HGD-Real	96.46	97.20
	LeapGestRecog	**100**	**100**

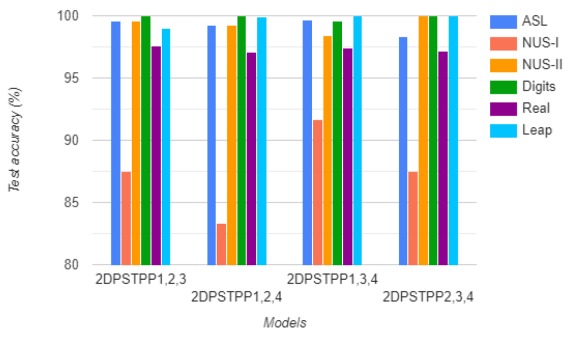

FIGURE 7. Histogram of result on NUS-II and HGD datasets with three pyramid pooling layers.

TABLE 6. The performance of the 2DPSTPP-Net model (test and train accuracy (%)) considering two pooling layers extracted from different convolutional blocks using NUS-II and HGD datasets.

Methods	Datasets	Train accuracy (%)	Test accuracy (%)
$2DPSTPP_{1,2}$	NUS-II	95.70	**96.48**
	HGD-Real	97.02	96.81
$2DPSTPP_{1,3}$	NUS-II	95.70	95.70
	HGD-Real	97.65	**97.66**
$2DPSTPP_{1,4}$	NUS-II	94.14	93.35
	HGD-Real	**97.85**	96.99
$2DPSTPP_{2,3}$	NUS-II	**96.09**	96.09
	HGD-Real	97.19	96.58
$2DPSTPP_{2,4}$	NUS-II	93.75	89.84
	HGD-Real	94.95	94.45
$2DPSTPP_{3,4}$	NUS-II	93.36	87.50
	HGD-Real	96.76	95.75

$2DPSTPP_{2,3,4}$. In addition, the test accuracy of NUS-I with 91.66% and HGD with 97.59% in subsets $2DPSTPP_{1,3,4}$ and $2DPSTPP_{1,2,3}$, respectively, are higher than other results.

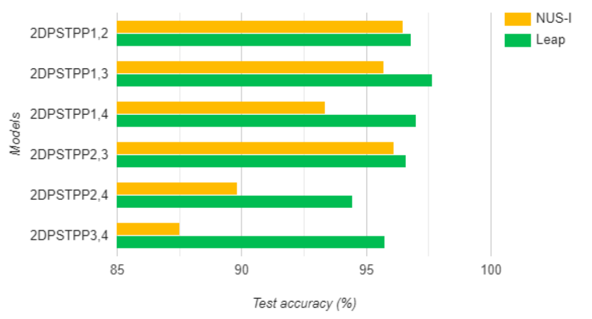

FIGURE 8. Histogram of result on NUS-II and HGD datasets with two pyramid pooling layers.

TABLE 7. The performance of the 2DPSTPP-Net model (test and train accuracy (%)) considering one pooling layer extracted from different convolutional blocks using NUS-II and HGD datasets.

Methods	Datasets	Train accuracy (%)	Test accuracy (%)
$2DPSTPP_1$	NUS-II	98.44	93.75
	HGD-Real	96.17	95.74
$2DPSTPP_2$	NUS-II	**98.83**	**96.61**
	HGD-Real	96.91	97.02
$2DPSTPP_3$	NUS-II	98.44	95.31
	HGD-Real	**98.18**	**97.86**
$2DPSTPP_4$	NUS-II	98.44	96.21
	HGD-Real	96.99	97.66

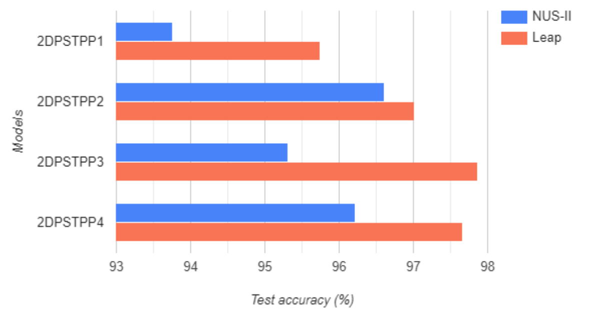

FIGURE 9. Histogram of result on NUS-II and HGD datasets with one pyramid pooling layers.

Table 6 and Figure 8 show the performance of 2DPSTPP-Net on six different subsets of two 2DPSTPP layers on two different NUS-II and HGD datasets. As is evident from Table 6 and Figure 8, NUS-II with 96.48% has highest test result in subset $2DPSTPP_{1,2}$ and HGD with 97.66% has the most accuracy in subset $2DPSTPP_{1,3}$. Moreover, as shown in Table 7 and the bar chart, which is obvious in Figure 9, has the highest test results among all subsets with one layer belonging to NUS-II with 99.61% in subset $2DPSTPP_2$ and HGD with 97.86% in subset $2DPSTPP_3$. As indicated in different tables and bar charts, the performance of Subset {1,2,3,4} is superior and the results achieved are much higher than other subsets.

As shown in Table 8, it can be seen that the CNN network with four blocks and four layers improves nearly 2.86% over VGG-13 on the NUS-II dataset and approximately 0.51% on the HGD-Real dataset. The result of this model is also improved by around 15.75%, 19.27%, 4.81% over VGG-16, VGG-11, and AlexNet on the NUS-II dataset, and 1.06%, 6.93%, 6.13% on the HGD-Real dataset, respectively. Hence, CNN is a suitable model to choose as the base model for our designed 2DPSTPP-Net method.

TABLE 8. The performance of other methods (test and train accuracy (%)) on NUS-II and HGD datasets.

Methods	Datasets	Train accuracy (%)	Test accuracy (%)
AlexNet [39]	NUS-II	90.62	91.02
	HGD-Real	87.70	89.89
VGG-11 [40]	NUS-II	80.86	76.56
	HGD-Real	87.64	89.09
VGG-13 [40]	NUS-II	92.58	92.97
	HGD-Real	94.56	95.51
VGG-16 [40]	NUS-II	76.17	80.08
	HGD-Real	94.13	94.96
CNN	NUS-II	**95.83**	**95.83**
	HGD-Real	**96.38**	**96.02**

V. CONCLUSION

We presented a novel method for hand gesture recognition. A convolutional neural network with four 2DPSTPP layers was considered for vision-based hand gesture detection to alleviate the problems of computing parameters and training complexity. Apart from this, the proposed method enabled distinguishing postures better, given complex backgrounds. Utilizing max-pooling in the 2DPSTPP layer improved the identification of relationships between different parts of objects. American sign language, the NUS Hand Posture dataset I, the NUS Hand Posture dataset II, the digits, the hand gesture datasets, and the hand gesture recognition database were used in our experiments to evaluate the results. The experiments showed that extracting 2DPSTPP features improves the recognition of hand gestures, especially in images with complex scenes.

REFERENCES

[1] J. P. Sahoo, A. J. Prakash, P. Pławiak, and S. Samantray, "Real-time hand gesture recognition using fine-tuned convolutional neural network," *Sensors*, vol. 22, no. 3, p. 706, Jan. 2022.
[2] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, "Robust part-based hand gesture recognition using Kinect sensor," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1110–1120, Aug. 2013.
[3] J. P. Wachs, "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, pp. 60–71, Dec. 2011.
[4] S. Lian, W. Hu, and K. Wang, "Automatic user state recognition for hand gesture based low-cost television control system," *IEEE Trans. Consum. Electron.*, vol. 60, no. 1, pp. 107–115, Feb. 2014.
[5] C. Wang, Z. Liu, and S.-C. Chan, "Superpixel-based hand gesture recognition with Kinect depth camera," *IEEE Trans. Multimedia*, vol. 17, no. 1, pp. 29–39, Jan. 2015.
[6] C. D. Sai Nikhil, C. U. Someswara Rao, E. Brumancia, K. Indira, T. Anandhi, and P. Ajitha, "Finger recognition and gesture based virtual keyboard," in *Proc. 5th Int. Conf. Commun. Electron. Syst. (ICCES)*, Jun. 2020, pp. 1321–1324.
[7] M. A. Ashok, M. P. Revathi, and B. Nirmal, "Finger recognition and gesture based virtual keyboard," *Turkish J. Comput. Math. Educ. (TURCOMAT)*, vol. 11, no. 3, pp. 2237–2245, 2020.
[8] (2018). *ASL Alphabet*. [Online]. Available: https://www.kaggle.com/datasets/grassknoted/asl-alphabet

[9] P. Pramod Kumar, P. Vadakkepat, and L. A. Poh, "The NUS hand posture datasets I," Nat. Univ. Singapore, Singapore, Tech. Rep., 2017. [Online]. Available: https://doi.org/10.25540/6PR9-R5HS
[10] P. P. Kumar, P. Vadakkepat, and L. A. Poh. "The NUS hand posture datasets II," Nat. Univ. Singapore, Singapore, Tech. Rep., 2017. [Online]. Available: https://doi.org/10.25540/6PR9-R5HS
[11] (2017). *Sign Language Digits Dataset*. [Online]. Available: https://www.kaggle.com/datasets/ardamavi/sign-language-digits-dataset
[12] (May 7, 2023). *Hand Gestures Dataset*. [Online]. Available: https://www.dlsi.ua.es/~jgallego/datasets/gestures
[13] T. Mantecón, C. R. del Blanco, F. Jaureguizar, and N. García, "Hand gesture recognition using infrared imagery provided by leap motion controller," in *Proc. Int. Conf. Adv. Concepts Intell. Vis. Syst. (ACIVS)*, Lecce, Italy, Oct. 2016, pp. 47–57, doi: 10.1007/978-3-319-48680-2_5.
[14] C.-J. Liao, S.-F. Su, and M.-C. Chen, "Vision-based hand gesture recognition system for a dynamic and complicated environment," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 2891–2895.
[15] C. Dewi and H. J. Christanto, "Combination of deep cross-stage partial network and spatial pyramid pooling for automatic hand detection," *Big Data Cognit. Comput.*, vol. 6, no. 3, p. 85, Aug. 2022.
[16] J. Nagi, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *Proc. IEEE Int. Conf. Signal Image Process. Appl. (ICSIPA)*, Nov. 2011, pp. 342–347.
[17] Y. S. Tan, K. M. Lim, C. Tee, C. P. Lee, and C. Y. Low, "Convolutional neural network with spatial pyramid pooling for hand gesture recognition," *Neural Comput. Appl.*, vol. 33, no. 10, pp. 5339–5351, 2021.
[18] A. Ashiquzzaman, H. Lee, K. Kim, H.-Y. Kim, J. Park, and J. Kim, "Compact spatial pyramid pooling deep convolutional neural network based hand gestures decoder," *Appl. Sci.*, vol. 10, no. 21, p. 7898, Nov. 2020.
[19] W. Zhou and K. Chen, "A lightweight hand gesture recognition in complex backgrounds," *Displays*, vol. 74, Sep. 2022, Art. no. 102226.
[20] K. Yang, R. Li, P. Qiao, Q. Wang, D. Li, and Y. Dou, "Temporal pyramid relation network for video-based gesture recognition," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 3104–3108.
[21] G. Zhu, L. Zhang, P. Shen, and J. Song, "Multimodal gesture recognition using 3-D convolution and convolutional LSTM," *IEEE Access*, vol. 5, pp. 4517–4524, 2017.
[22] G. Jianchun, G. Jiannuan, and W. Lili, "Gesture recognition method based on attention mechanism for complex background," *J. Phys., Conf. Ser.*, vol. 1873, no. 1, Apr. 2021, Art. no. 012009.
[23] P. K. Pisharady, P. Vadakkepat, and A. P. Loh, "Attention based detection and recognition of hand postures against complex backgrounds," *Int. J. Comput. Vis.*, vol. 101, no. 3, pp. 403–419, Feb. 2013.
[24] H. Wang, P. Wang, Z. Song, and W. Li, "Large-scale multimodal gesture recognition using heterogeneous networks," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 3129–3137.
[25] A. Dadashzadeh, A. T. Targhi, M. Tahmasbi, and M. Mirmehdi, "HGR-net: A fusion network for hand gesture segmentation and recognition," *IET Comput. Vis.*, vol. 13, no. 8, pp. 700–707, Dec. 2019.
[26] Z. Cui, Y. Lei, Y. Wang, W. Yang, and J. Qi, "Hand gesture segmentation against complex background based on improved atrous spatial pyramid pooling," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 9, pp. 11795–11807, Sep. 2023.
[27] G. Zhang, L. Wang, L. Wang, and Z. Chen, "Hand-raising gesture detection in classroom with spatial context augmentation and dilated convolution," *Comput. Graph.*, vol. 110, pp. 151–161, Feb. 2023.
[28] V. Jain, A. Jain, A. Chauhan, S. S. Kotla, and A. Gautam, "American sign language recognition using support vector machine and convolutional neural network," *Int. J. Inf. Technol.*, vol. 13, no. 3, pp. 1193–1200, Jun. 2021.
[29] S. Sharma and S. Singh, "Vision-based hand gesture recognition using deep learning for the interpretation of sign language," *Exp. Syst. Appl.*, vol. 182, Nov. 2021, Art. no. 115657.
[30] M. Al-Hammadi, G. Muhammad, W. Abdul, M. Alsulaiman, M. A. Bencherif, T. S. Alrayes, H. Mathkour, and M. A. Mekhtiche, "Deep learning-based approach for sign language gesture recognition with efficient hand gesture representation," *IEEE Access*, vol. 8, pp. 192527–192542, 2020.
[31] A. S. M. Miah, M. A. M. Hasan, J. Shin, Y. Okuyama, and Y. Tomioka, "Multistage spatial attention-based neural network for hand gesture recognition," *Computers*, vol. 12, no. 1, p. 13, Jan. 2023.
[32] A. S. M. Miah, Md. A. M. Hasan, and J. Shin, "Dynamic hand gesture recognition using multi-branch attention based graph and general deep learning model," *IEEE Access*, vol. 11, pp. 4703–4716, 2023.
[33] A. A. Q. Mohammed, J. Lv, M. S. Islam, and Y. Sang, "Multi-model ensemble gesture recognition network for high-accuracy dynamic hand gesture recognition," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 6, pp. 6829–6842, Jun. 2023.
[34] M. M. Damaneh, F. Mohanna, and P. Jafari, "Static hand gesture recognition in sign language based on convolutional neural network with feature extraction method using ORB descriptor and Gabor filter," *Exp. Syst. Appl.*, vol. 211, Jan. 2023, Art. no. 118559.
[35] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "DenseASPP for semantic segmentation in street scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3684–3692.
[36] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[37] Y. Song, L. Bliek, T. Xia, and Y. Zhang, "A temporal pyramid pooling-based convolutional neural network for remaining useful life prediction," in *Proc. 31st Eur. Saf. Rel. Conf. (ESREL)*, 2021, pp. 810–817.
[38] A. Paszke, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS-W*, 2017, pp. 1–4.
[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.
[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

FARZANEH JAFARI received the M.Sc. degree in computer science from the University of Tehran, Tehran, Iran. She is currently pursuing the Ph.D. degree in computing science with the University of Alberta, Edmonton, Canada. She was a Research Assistant with the University of Tehran, the K. N. Toosi University of Technology, and the Iran Polymer and Petrochemical Institute, Tehran.

ANUP BASU received the Ph.D. degree in computer science from the University of Maryland, College Park, USA. He originated the use of foveation for image, video, stereo, and graphics communication in the early 1990's; an approach that is now widely used in industrial standards. He pioneered the active camera calibration method emulating the way the human eyes work and showed that this method is far superior to any other camera calibration method. He pioneered a single-camera panoramic stereo and several new approaches merging foveation and stereo with application to 3D TV visualization and better depth estimation. His current research interests include multi-dimensional image processing and visualization for medical, consumer, remote sensing applications, multimedia in education and games, and robust wireless 3D multimedia transmission. He has been a Professor with the CS Department, UofA, since 1999. He has also held the following positions: a Visiting Professor with the University of California at Riverside, Riverside, from 2003 to 2004; a Guest Professor with the Technical University of Austria, Graz, in 1996; and the Director of the Hewlett-Packard Imaging Systems Instructional Laboratory, UofA, from 1997 to 2000. He is a fellow of the American Neurological Association. He has also been the NSERC, iCORE, and Castle Rock Research Chair.