Neural Computing and Applications (2021) 33:5339–5351
https://doi.org/10.1007/s00521-020-05337-0

ORIGINAL ARTICLE

Convolutional neural network with spatial pyramid pooling for hand gesture recognition

Yong Soon Tan[1] · Kian Ming Lim[1] · Connie Tee[1] · Chin Poo Lee[1] · Cheng Yaw Low[1]

Received: 22 October 2019 / Accepted: 2 September 2020 / Published online: 15 September 2020

Abstract
Hand gesture provides a means for human to interact through a series of gestures. While hand gesture plays a significant role in human–computer interaction, it also breaks down the communication barrier and simplifies communication process between the general public and the hearing-impaired community. This paper outlines a convolutional neural network (CNN) integrated with spatial pyramid pooling (SPP), dubbed CNN–SPP, for vision-based hand gesture recognition. SPP is discerned mitigating the problem found in conventional pooling by having multi-level pooling stacked together to extend the features being fed into a fully connected layer. Provided with inputs of varying sizes, SPP also yields a fixed-length feature representation. Extensive experiments have been conducted to scrutinize the CNN–SPP performance on two well-known American sign language (ASL) datasets and one NUS hand gesture dataset. Our empirical results disclose that CNN–SPP prevails over other deep learning-driven instances.

Keywords Convolutional neural network (CNN) · Spatial pyramid pooling (SPP) · Hand gesture recognition · Sign language recognition

1 Introduction

Hand gesture recognition not only has the ability to transform human–computer interaction (HCI) but also simplifies communication with each other, especially the communication process between the deaf community with the general public. People with deafness communicate with others primarily through a language of hand gesture, formally known as sign language. Despite its well-known existence, only minority of the general public have the knowledge to interpret sign language. Hand gesture recognition system breaks down this communication barrier by allowing people to understand the meaning behind different sign language gestures effortlessly. There are many challenges in vision-based hand gesture recognition, for example, hand size variation, skin tone and color, illumination, view point variation, similarity in different gestures and the complex natural background. In short, recognizing static hand gesture in images can be very challenging due to the diverse and varied conditions presented in the images. In recent years, many works have been carried out to address these problems.

2 Related works

Prior to the success of deep learning, hand-crafted feature extraction methods with classifier were much more popular, and widely adopted approach. Hand-crafted feature extraction methods extract useful features from images, and the extracted features are then classified by the classifier. This approach has proven useful in tackling various image classification tasks, and many researchers have adopted the same approach for hand gesture recognition [1–3].

In [4], a novel feature extraction method was proposed to calculate the landmark distance of line joining a point on the boundary of the hand to its centroid. On the other hand,

✉ Kian Ming Lim
kmlim@mmu.edu.my

Yong Soon Tan
tanygsn@gmail.com

[1] Faculty of Information Science and Technology (FIST), Multimedia University, Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia

the work in [5] introduced speeded up robust features (SURF) to extract prominent features from the hand gesture images. Both methods utilized *k*-nearest neighbor (K-NN) as the classifier. Besides, local sparse representation classification (LSRC) with L1-norm was applied in [6] to address the problem of computational complexity while retaining similar and better result against comparable methods. Furthermore, support vector machine (SVM) was used to classify local binary pattern (LBP) features in [7], Zernike moments, histograms of oriented gradients (HOG) and LBP in [8] and global and local features in [9]. Additionally, [10] extracted HOG features from images and these features were classified by Adaptive Boost (AdaBoost).

Artificial neural network (ANN) was used in [11] with discrete cosine transform (DCT) and moment invariant as the feature extractor. In [12], the features were extracted by Fourier coefficients amplitude. Random line segment was proposed for extracting feature in [13], and the features were subsequently classified by random forest (RF). In addition, HOG and Zernike moments were used to describe the features in [14], and deep belief network (DBN) with three restricted Boltzmann machines (RBMs) was used for ASL recognition.

Since the reintroduction of convolutional neural network (CNN) dubbed as AlexNet in [15], deep learning has gained huge momentum and popularity among the machine learning researchers. Since then, CNN has been in the spotlight and applied in a wide range of applications in image classification tasks, including hand gesture recognition [16]. The popularity is mainly due to its innate ability to discover complex patterns in data and eliminate the need for manually constructing hand-crafted feature or weighing which method to use.

Recently, numerous CNN variants were proposed for hand gesture recognition, [17] achieved 91.26% accuracy for Thai finger spelling, and [18] obtained 73.4% and 98.5% accuracy for HUST-ASL dataset and NTU-HD dataset, respectively. On the other hand, [19, 20] obtained 96.1% and 93% accuracy for Japanese sign language (JSL), respectively, while [21, 22] both achieved 93.3% accuracy for ASL.

Besides that, [23] attained 99% for Irish sign language (ISL), while [24, 25] attained 96.2% and 84.5% accuracy for alphabet of sign language of Peru (LSP); [26] achieved 84.5% on self-constructed dataset with ten gestures. In addition to that, [27] proposed Growing-When-Required (GWR) network, which is an unsupervised-learning based network for Cyrillic finger spelling and obtained 93% accuracy.

While some aforementioned CNN variants were able to obtain satisfactory recognition rate on their respective datasets, none of them were tested or able to scale well across multiple benchmark datasets and achieved the kind of recognition rate essential to real-life use cases. Hence, this paper proposed a new variant of CNN tested on three benchmark datasets.

The main contribution of this paper is of threefold:

1. A synergistic network construction integrating convolutional neural network with spatial pyramid pooling (CNN–SPP) is devised for vision-based hand gesture recognition.
2. Unlike the typical max-pooling layer, SPP derives a set of fixed-length feature through multi-level feature summarization. We affix an individual SPP operator to each convolutional block in this paper to render a composite feature for the subsequent fully connected layer. In doing so, it helps the gradients flow directly to the earlier layers, thereby facilitating them to learn better. In principle, SPP is interpretable as a feature compressor forwarding a generally more compact representation for learning the ultimate softmax classifier.
3. Due to limited training data accessible for network training, a summation of nine augmentation techniques are applied to learn a high-complexity network for classification task.

The TensorFlow implementation of CNN–SPP will be released upon the acceptance of this paper.

3 Proposed solution

Figure 1 portrays the network construction for the proposed CNN–SPP. There are two convolutional blocks (conv-block), of which each conv-block consists of $L = 4$ convolutional layers. Each layer in conv-block 1 is trained with $3 \times 3 \times 64$ filters, while each layer in conv-block 2 is trained with $3 \times 3 \times 128$ filters. The SPP features are extracted from the Lth convolutional layer in each conv-block. Subsequently, the SPP features are vectorized and concatenated to form a single vector as the input for the first fully connected layer.

3.1 Convolutional neural network with spatial pyramid pooling (CNN–SPP)

As of other CNNs, the fundamental building block of CNN–SPP is the conv-block consisting of multiple convolutional layers with zero padding, batch normalization and rectified linear unit (ReLU) activation.

In convolutional layer, filters are used as the sliding window as it slides through the whole image to compute the output, which is commonly referred to as the feature maps. In practice, zero padding is frequently used in the

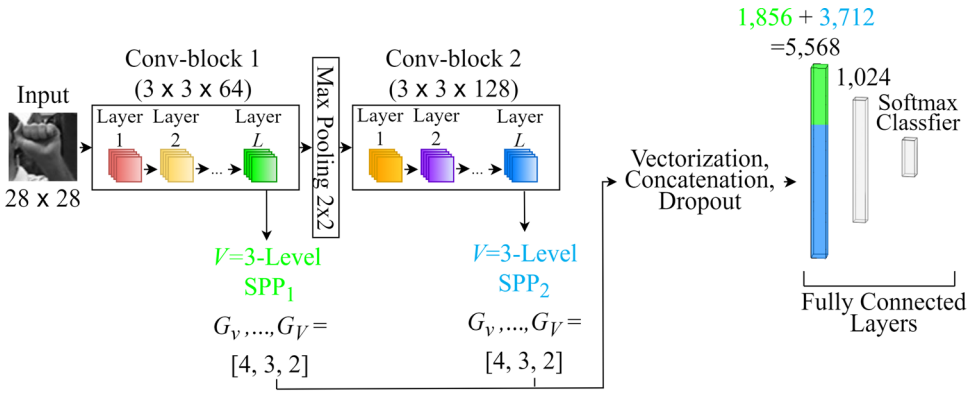

Fig. 1 The proposed CNN–SPP network architecture, which consists of two conv-blocks, each with $L = 4$ layers, two SPP operations are applied to the feature maps of Lth convolutional layer from each conv-block, followed by two fully connected layers, which includes a softmax classifier

convolution operation. As some of the useful features like curves and edges are located at the edge of the image, zero padding is performed to allow the filters to extract useful information at the edge of the image. In addition, convolution without padding not only shrinks the size of output feature maps, but also tends to lose information at the edge of the image. Moreover, discarding the information at the edge is not ideal for detecting features, as some of the features like the contour of the hand or edges of fingers might be located at the edge of an image. This will affect the subsequent convolutional layer's ability to learn useful features.

Backpropagation allows filters in the convolutional layer to learn by updating its weights after each iteration in order to minimize the loss incurred by the predicted output. CNN–SPP consists of two conv-blocks, each conv-block with $L = 4$ convolutional layers. Each layer is followed by another layer, with no pooling layer in between layers in conv-block. Max pooling is performed on the feature maps of the Lth convolutional layer of conv-block 1 to reduce computational complexity.

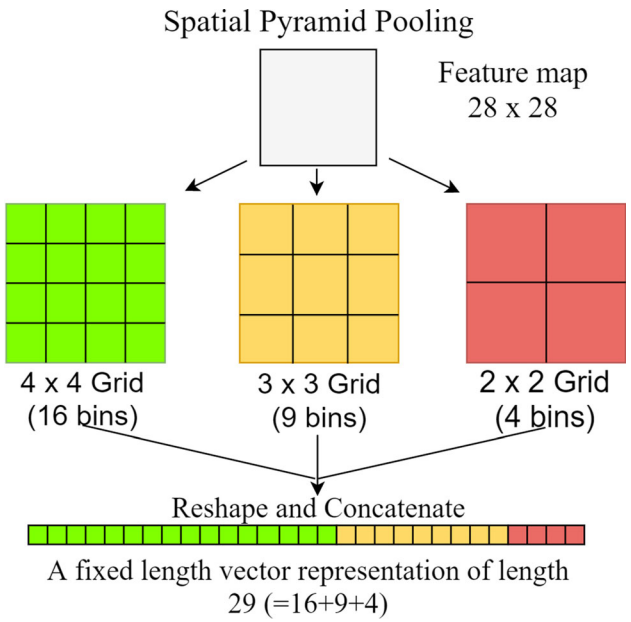

Fig. 2 A V-level SPP instance with grid size $[G_1, G_2, \ldots, G_V]$. In this example, we set $V = 3$, $G_1 = 4, G_2 = 3, G_3 = 2$, producing a fixed-length vector of 29 features given a single feature map as input

Batch normalization [28] is utilized to reduce internal covariance shift. To put it bluntly, batch normalization normalizes the feature maps and allows the subsequent layers to learn better by having the same mean and variance. The mean and variance are governed by β and γ values, which are learnt by the network through backpropagation. Batch normalization helps to limit the effect of changes in the distribution of values. The effect happens during training when the network updates the parameters in the earlier layers, which the latter layer has to learn on. This, in turn, helps each layer to learn more independently of the other layers and in the meantime speeds up learning.

With a d-dimensional input, $\boldsymbol{x} = (x^{(1)} \ldots x^{(d)})$, each dimension will be normalized by

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}} \tag{1}$$

where $x^{(k)}$ denotes a particular activation, $E[x^{(k)}] = \frac{1}{m}\sum_{i=1}^{m} x_i^{(k)}$ and $\sqrt{\mathrm{Var}[x^{(k)}]} = \frac{1}{m}\sum_{i=1}^{m}(x_i^{(k)} - E[x^{(k)}])^2 + \epsilon$. Here, $\epsilon = 0.001$ is added to the equation for numerical stability, by ensuring $\sqrt{Var[x^{(k)}]} > 0$. For each activation $\hat{x}^{(k)}$, a pair of scaling and shifting parameters, $\gamma^{(k)}$ and $\beta^{(k)}$, scale and shift the normalized value by

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}. \tag{2}$$

These parameters are learnt through backpropagation. The original activation, $x^{(k)}$, can be recovered, if the network learns, that is, the optimal thing to do through backpropagation, by setting $\gamma^{(k)} = \sqrt{Var[x^{(k)}]}$ and $\beta^{(k)} = E[x^{(k)}]$.

3.2 Spatial pyramid pooling (SPP)

Typically, the conventional architecture of CNN affixes a pooling layer after one or multiple convolutional layers.

Fig. 3 Sample images from ASL dataset, where each row denotes a single-hand gesture set contributed by a signer. (Adapted from [31])

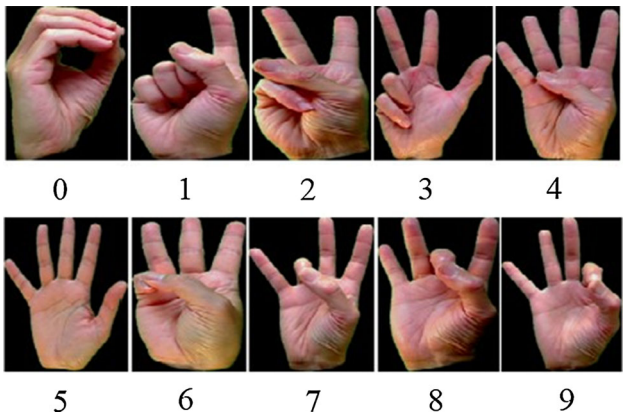

Fig. 4 Sample images from ASL with digits dataset

The pooling layer aims to scale down the spatial size of the feature maps which in turn reduces the number of training parameters and computational complexity.

Pooling is usually done in a non-overlapping fashion. As the filter slides through the feature maps in stride, it chooses a single value to be the representation of that particular spatial location. Max pooling extracts the biggest value in the spatial location, while mean pooling takes the average of all pixel values in the spatial location. Although pooling reduces the computational complexity, it loses information that is needed to detect precise relationships between the object parts.

On the contrary, spatial pyramid pooling (SPP) [29] alleviates the problem by having multiple pooling operations stacked together, each with different filter sizes. SPP captures more spatial information than a conventional pooling operation simply by having more spatial bins, and this allows the network to better distinguish among the complex data. Another upside of SPP is that it produces a fixed-size output regardless of the input size.

Let h and w denote the height and width of a feature map, respectively, with a SPP grid size of $G \times G$, the filter size denoted by $f = f_{\mathrm{h}} = f_{\mathrm{w}}$, is calculated as $f_{\mathrm{h}} = \lceil h/G \rceil$, and $f_{\mathrm{w}} = \lceil w/G \rceil$. The $\lceil . \rceil$ denotes the ceiling operation. Stride height and stride width for pooling are denoted by S_{h} and S_{w}, respectively, where $S_{\mathrm{h}} = \lceil h/G \rceil$ and $S_{\mathrm{w}} = \lceil w/G \rceil$. Padding of the feature map is required, where $pad_{\mathrm{h}} = G \times S_{\mathrm{h}} - h$ denotes padding for height and $pad_{\mathrm{w}} = G \times S_{\mathrm{w}} - w$ denotes padding for width.

Fig. 5 Sample images from NUS hand gesture dataset

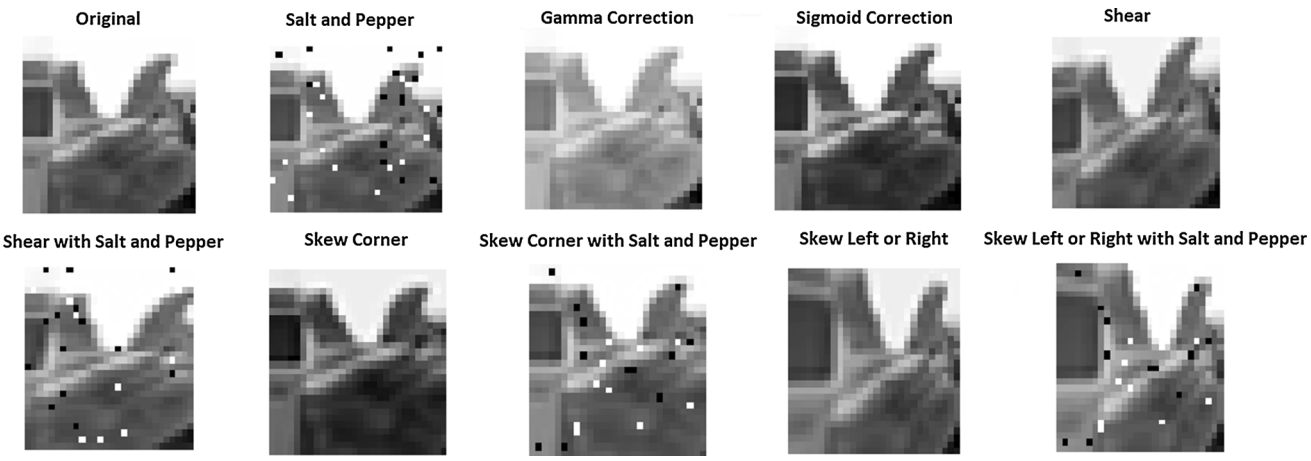

Fig. 6 Original image with nine other augmented images

Table 1 Cross-validation results for CNN–SPP, with and without augmented data (AD)

Dataset	Cross-validation set k =fivefold cross-validation	Test accuracy (%) (without AD)	Test accuracy (%) (with AD)
ASL	1	99.90	99.99
	2	99.95	100
	3	99.98	100
	4	99.89	100
	5	99.96	100
	Average	99.94	99.99
ASL with digits	1	99.20	100
	2	99.40	99.60
	3	99.20	99.60
	4	98.81	99.20
	5	99.20	99.80
	Average	99.17	99.64
NUS hand gesture	1	95.75	98.50
	2	95.50	99.25
	3	97.25	97.75
	4	96.25	98.25
	5	95.00	98.25
	Average	95.95	98.40

Each SPP operation yields a fixed-length feature representation. Let $v = 1, \ldots, V$ denote V-level of SPP, provided with a single feature map of any dimension, the resulted feature length, denoted by spp, for V-level SPP operation is determined by Eq. 3. The v in G_v denotes the index of SPP grid size, $G_v \times G_v$, and $G_v = \mathbb{R} > 0$. On the other hand, Eq. 4 determines the total feature length yielded by SPP applied on Lth layer of conv-block i, denoted by SPP_i. The fm_L denotes the total number of feature maps of Lth layer of conv-block i. Figure 2 illustrates SPP with V =three-level pooling, with $G = [4, 3, 2]$.

$$\text{spp} = \sum_{v=1}^{V} G_v^2 \tag{3}$$

$$\text{SPP}_i = \text{spp} \times fm_L. \tag{4}$$

In this work, we vectorize and concatenate the SPP features from Lth convolutional layer's feature maps in conv-block 1 and conv-block 2. Extracting SPP features from earlier layer not only extends the features being forwarded to the fully connected layer, but also helps the gradients flow directly into the earlier layers and hence makes the earlier

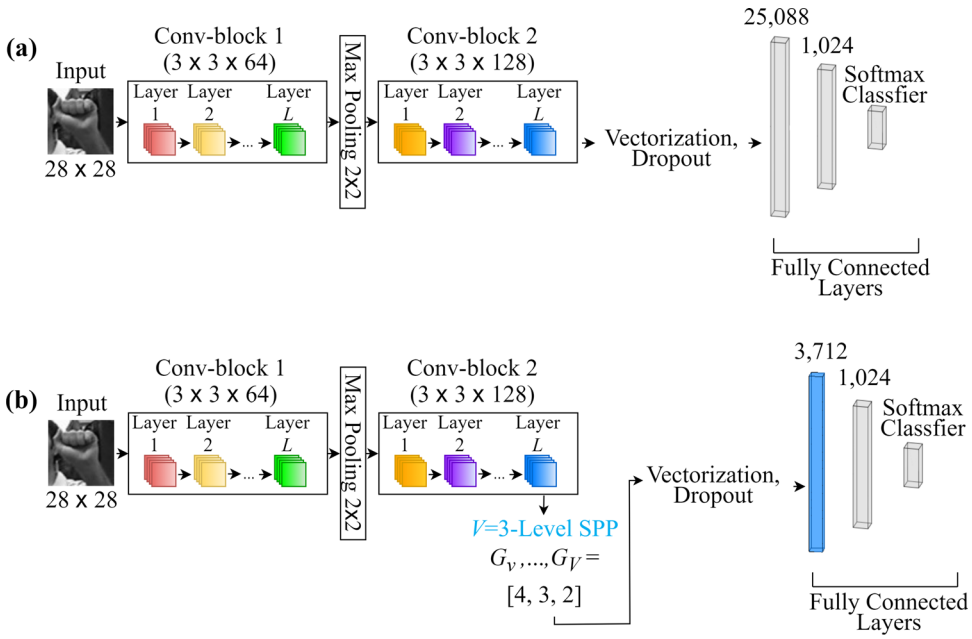

Fig. 7 Network architecture of **a** typical CNN and **b** CNN–SPP-1. Conv-block 2 yields a tensor of $14 \times 14 \times 128$, typical CNN produces a flatten feature of $25,088 (= 14 \times 14 \times 128)$ dimensions, whereas CNN–SPP-1 produces a flattened feature of $3,712 (= (4^2 + 3^2 + 2^2) \times 128)$ dimensions for fully connected layer

Table 2 Performance comparison between CNN with no SPP, with only one SPP (CNN–SPP-1) and with two SPPs (CNN–SPP), in terms of test accuracy (%), with and without augmented data (AD)

Method	Dataset	Test accuracy (%) (without AD)	Test accuracy (%) (with AD)	Average inference time (milliseconds per image)
Typical CNN (with no SPP)	ASL	99.91	99.99	4.1
	ASL with digits	98.69	99.44	
	NUS hand gesture	92.60	97.65	
	Average	97.07	99.03	
CNN–SPP-1 (with one SPP)	ASL	99.94	99.99	4.1
	ASL with digits	99.01	99.44	
	NUS hand gesture	94.15	97.25	
	Average	97.70	98.89	
CNN–SPP (with two SPPs)	ASL	99.94	99.99	4.5
	ASL with digits	99.17	99.64	
	NUS hand gesture	95.95	98.40	
	Average	98.35	99.34	

layers learn better. Subsequent to that, the concatenated SPP features are served as input of the first fully connected layer. The first fully connected layer has 5568 neurons fully connected with 1024 neurons, and the second layer, which is also the output layer, consists of 1024 neurons fully connected with the number of neurons corresponds to the number of classes. A dropout rate of 50% is applied to the concatenated SPP features before feeding it into the first fully connected layer and also applied to the output of first fully connected layer. Besides, batch normalization is applied to the first fully connected layer. Dropout [30] helps by preventing the network from overfitting data by randomly turning off numerous neurons at a certain rate during training. This lessens the neurons dependency on each other, which, if left unprocessed, often leads to overfitting with the neurons strongly dependent on each other.

4 Experiments

We employ three datasets in our experiments, specifically the ASL finger spelling dataset [31], ASL finger spelling with digits dataset [32] and NUS hand gesture dataset [33]. ASL finger spelling dataset (referred to as ASL dataset) consists of 65,774 images for 24 classes of gesture. ASL

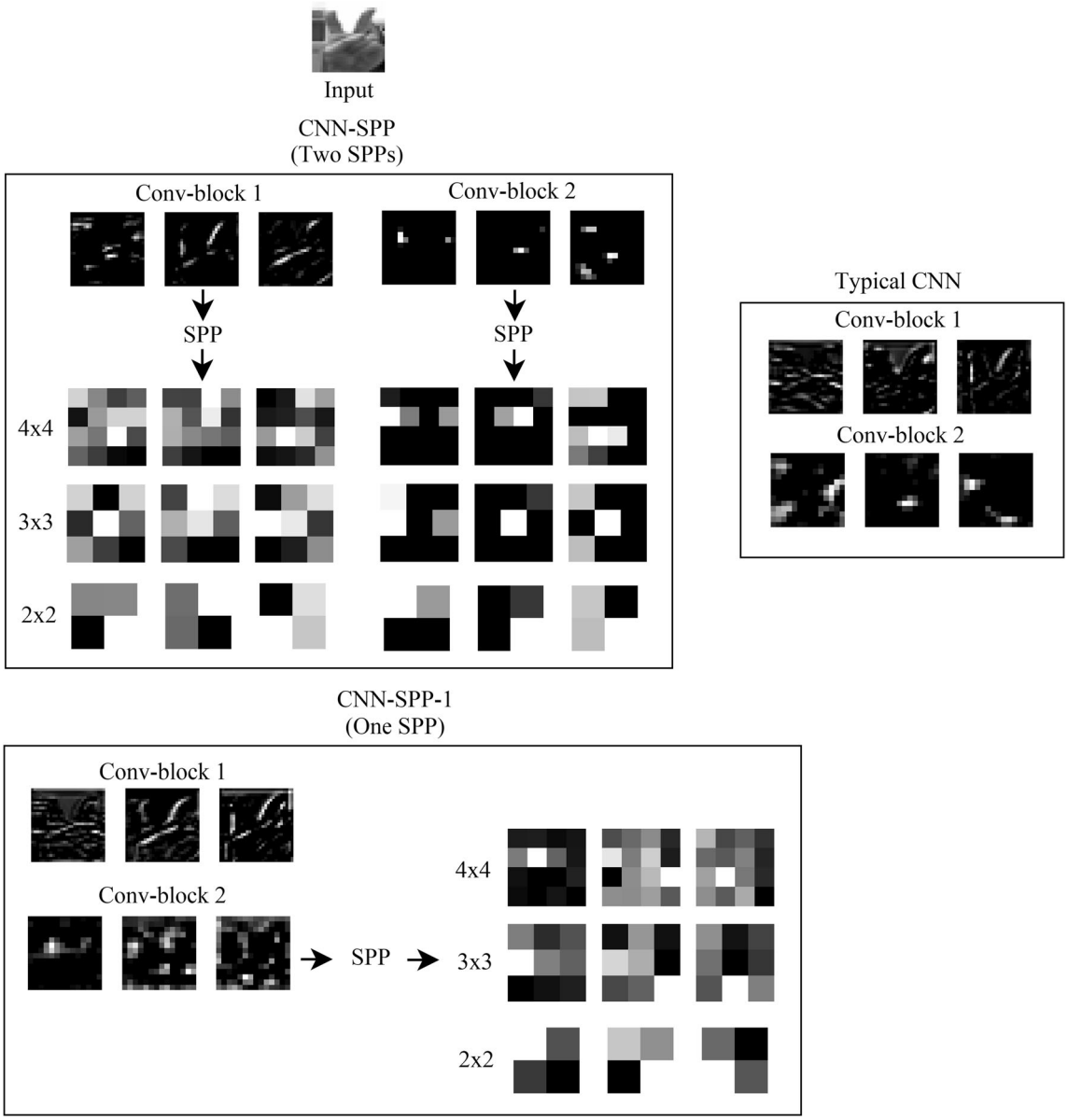

Fig. 8 Feature maps generated by conv-blocks in CNN–SPP, CNN–SPP-1 and typical CNN

finger spelling with digits dataset (referred to as ASL with digits dataset) comprises 36 classes of gesture and a total of 2515 images. NUS hand gesture dataset (referred to as NUS hand gesture dataset) consists of 2000 images for ten classes of gesture. Figures 3, 4 and 5 show the sample images for ASL dataset, ASL with digits dataset and NUS hand gesture dataset, respectively.

4.1 Experimental setup

We apply aggressive data augmentation to alleviate the problem of insufficient training data. In our experiments, every single image of the training sets from the aforementioned datasets is diversified by nine data augmentation techniques. These include shearing, salt and pepper, gamma correction, sigmoid correction and perspective transformation techniques, namely corner skewing and left or right skewing. Besides, there are other three combined manipulations, i.e., shearing with salt and pepper, corner skewing with salt and pepper and left or right skewing with salt and pepper. Figure 6 shows nine augmented images of the aforementioned augmentation techniques.

Following the testing paradigm used in numerous works on CNN for image classification tasks [34–36], as well as in hand gesture recognition [37, 38], we performed $k =$ fivefold cross-validation in our experiments for each dataset. In the experiments, the aforementioned data augmentation techniques are only applied to images in the

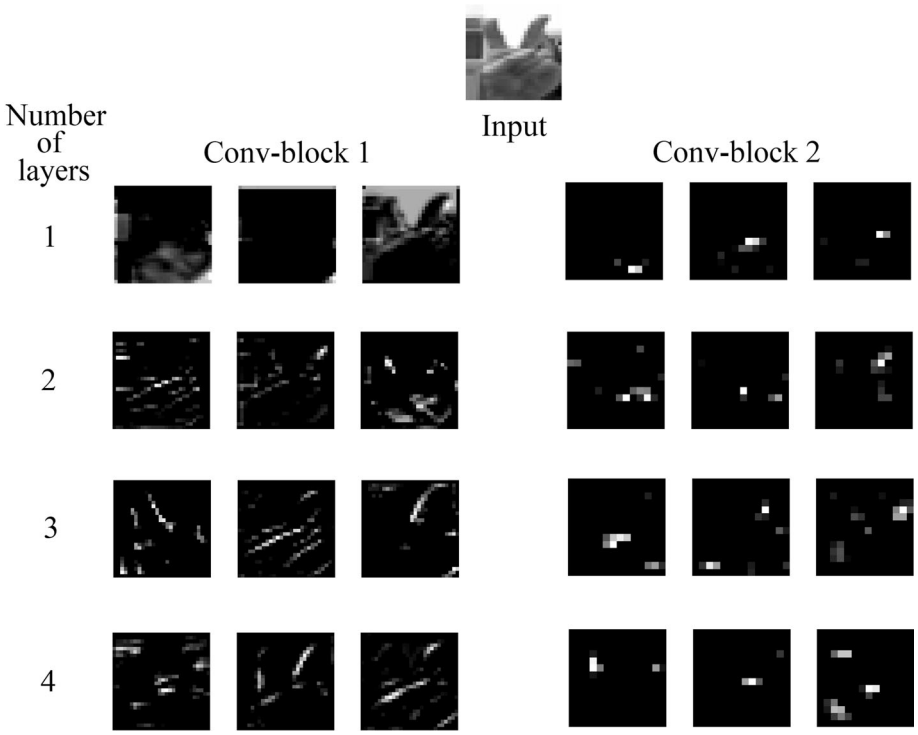

Fig. 9 Feature maps generated by conv-blocks with different L in CNN–SPP. Each row denotes different numbers of convolutional layers in the conv-blocks

training set, while the images in the testing set remained untouched. The CNN–SPP performance for all five folds is summarized in Table 1.

4.2 Experimental analysis

In this section, an ablation study was conducted and to demonstrate the effectiveness of having SPP in the network and the impact it has on the network's ability to generalize. Firstly, CNN–SPP, which utilizes SPP to extract features from Lth convolutional layer from conv-block 1 and conv-block 2 is tested. Secondly, a toned-down version of CNN–SPP referred to as CNN–SPP-1 (with one SPP) which utilizes SPP with grid $G = [4, 3, 2]$ is used to extract features from the Lth convolutional layer of conv-block 2 only. Finally, a typical CNN (with no SPP) is tested as well. Figure 7 illustrates the network architecture of typical CNN and CNN–SPP-1. Table 2 shows the results of typical CNN, CNN–SPP-1 and CNN–SPP, tested on datasets with and without augmented data. Time is also included in Table 2 to discern the difference in runtime during inference stage. All models test times are ran and captured on Tensorflow 1.15.2 with Tesla P100 GPU.

Judging by the results in Table 2, the effectiveness of CNN–SPP has been demonstrated to be superior. The proposed CNN–SPP achieves better recognition accuracy on all three datasets tested. While two other networks achieved similar recognition accuracy on ASL dataset and ASL with digits dataset, the gap of recognition accuracy widens on a more challenging NUS hand gesture dataset. This further validates the effectiveness of having two SPPs, especially the one from the earlier conv-block, which only account for 1, 856 neurons out of the total 5568 neurons of the input of first fully connected layer. As depicted in Fig. 7, typical CNN produces a flatten feature of 25,088 dimensions. In contrast, the proposed CNN–SPP (Fig. 1) produces a flattened feature of 5568 dimensions. This shows that SPP has tremendously reduced the dimension of the features by 80% . In terms of inference time, typical CNN and CNN–SPP-1 performed similarly, which required 4.1 milliseconds, while CNN–SPP took 4.5 milliseconds.

Moreover, Fig. 8 illustrates feature maps generated by conv-blocks in CNN–SPP (with two SPPs), CNN–SPP-1 and typical CNN. As shown in Fig. 8, conv-blocks predominantly output feature maps with only some regions of the feature map being illuminated, while other regions being rendered black. This coincides with the notion that having multiple consecutive convolutional layers suppresses irrelevant features, which is illustrated in Fig. 9.

Additionally, in comparison with CNN–SPP-1 and typical CNN, it is noted that conv-blocks in CNN–SPP output the least illuminated feature maps. This demonstrates the effect of performing SPP on lth layer in conv-block 1, which helps gradients flow directly back into earlier layers and thus allows earlier layers in the network to learn better and suppress irrelevant features.

Table 3 Performance comparison in terms of testing accuracy (%) for cases with and without augmented data (AD)

Method	Dataset	Test accuracy (%) (without AD)	Test accuracy (%) (with AD)
CNN baseline [24]	ASL	99.85	99.95
	ASL with digits	98.69	99.32
	NUS hand gesture	89.15	94.35
	Average	95.90	97.87
ADCNN [25]	ASL	98.50	99.28
	ASL with digits	98.49	98.97
	NUS hand gesture	83.10	89.30
	Average	93.36	95.85
DAG-CNN [26]	ASL	99.89	99.99
	ASL with digits	98.13	99.28
	NUS hand gesture	91.05	96.85
	Average	96.36	98.71
CNN–SPP	ASL	99.94	99.99
	ASL-with digits	99.17	99.64
	NUS hand gesture	95.95	98.40
	Average	98.35	99.34

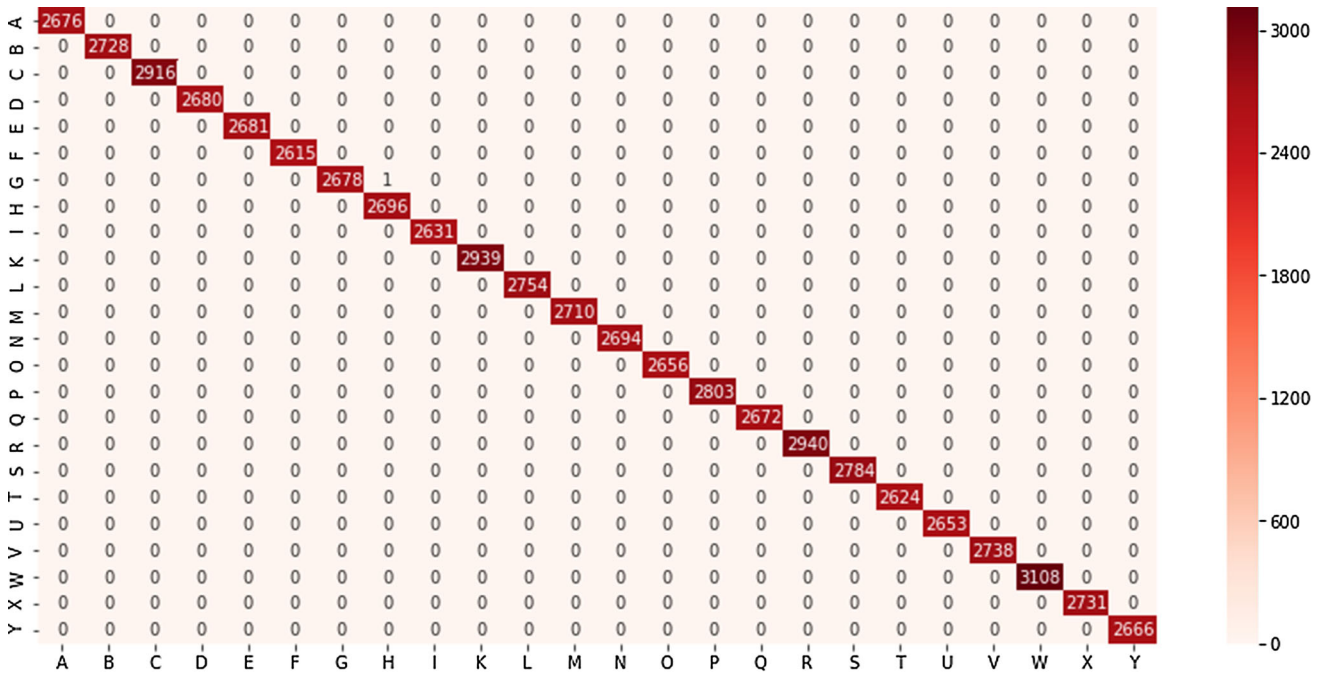

Fig. 10 Confusion matrix for ASL dataset (with data augmentation)

4.3 Experimental results and discussion

This section compares the CNN–SPP performance in hand gesture recognition to that of the state of the arts, including an AlexNet variant (termed as CNN baseline henceforward) [24], the adapted deep convolutional neural network (ADCNN) [25] and the direct acyclic graph-based convolutional neural network (DAG-CNN) [26], with and without data augmentation applied.

ADCNN [25] has two convolutional layers with He initialization. Each convolutional layer is immediately followed by a max pooling and a dropout rate of 20%. The fully connected layers consist of two layers, with 800 neurons fully connected with 128 neurons in the first layer

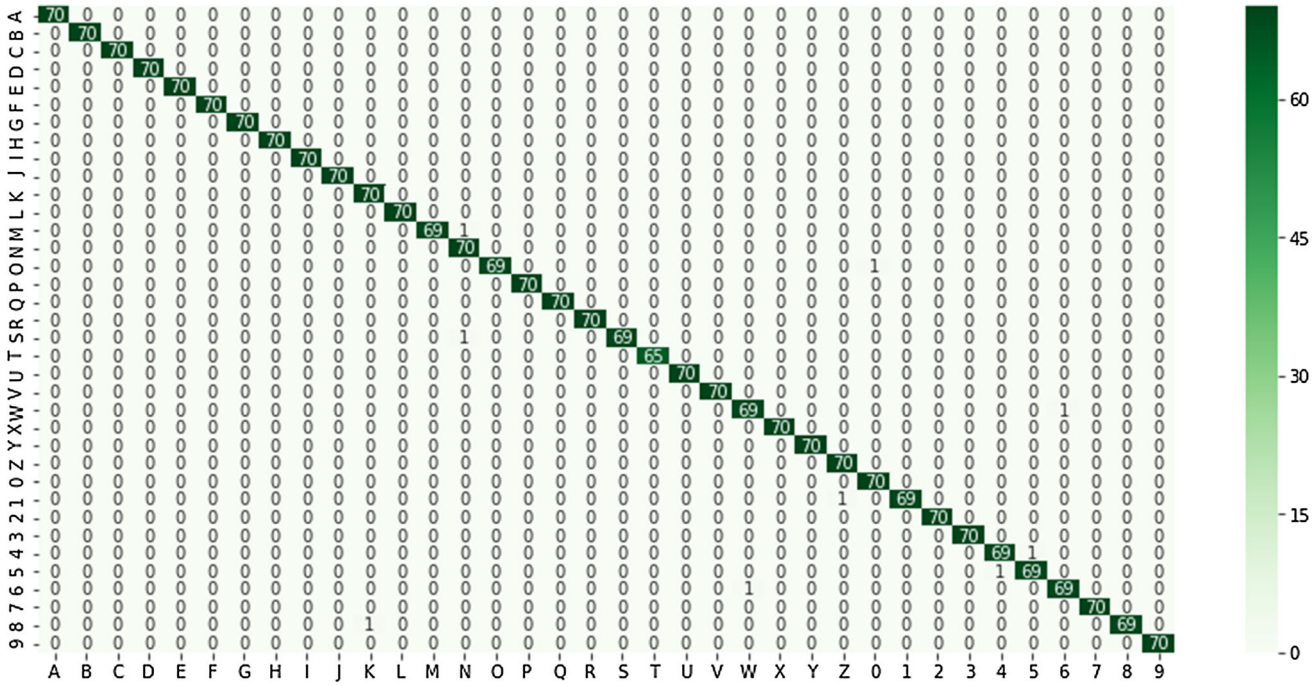

Fig. 11 Confusion matrix for ASL with digits dataset (with data augmentation)

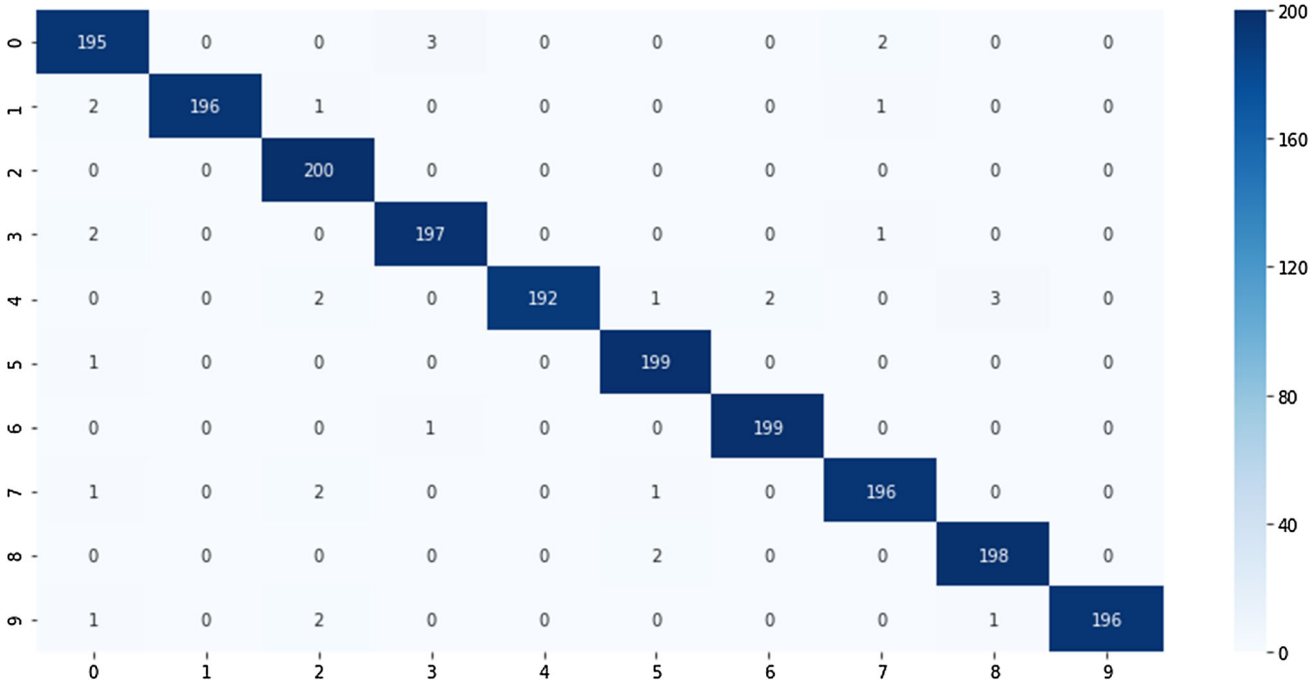

Fig. 12 Confusion matrix for NUS hand gesture dataset (with data augmentation)

and 128 neurons fully connected with 64 neurons in the second layer; Xavier initialization is used in the softmax layer. CNN baseline [24] has three convolutional layers, each immediately followed by a max-pooling operation; dropout is applied after the third max-pooling operation. As for fully connected layers, two layers are used, each with 4,096 neurons, and both apply a dropout rate of 50%. DAG-CNN [26] has two paths of the same layers layout, each path consists of five convolutional layers, but it is different in the number of feature maps used and three max-pooling operations in between convolutional layers. Two fully connected layers are used, with 256 neurons in

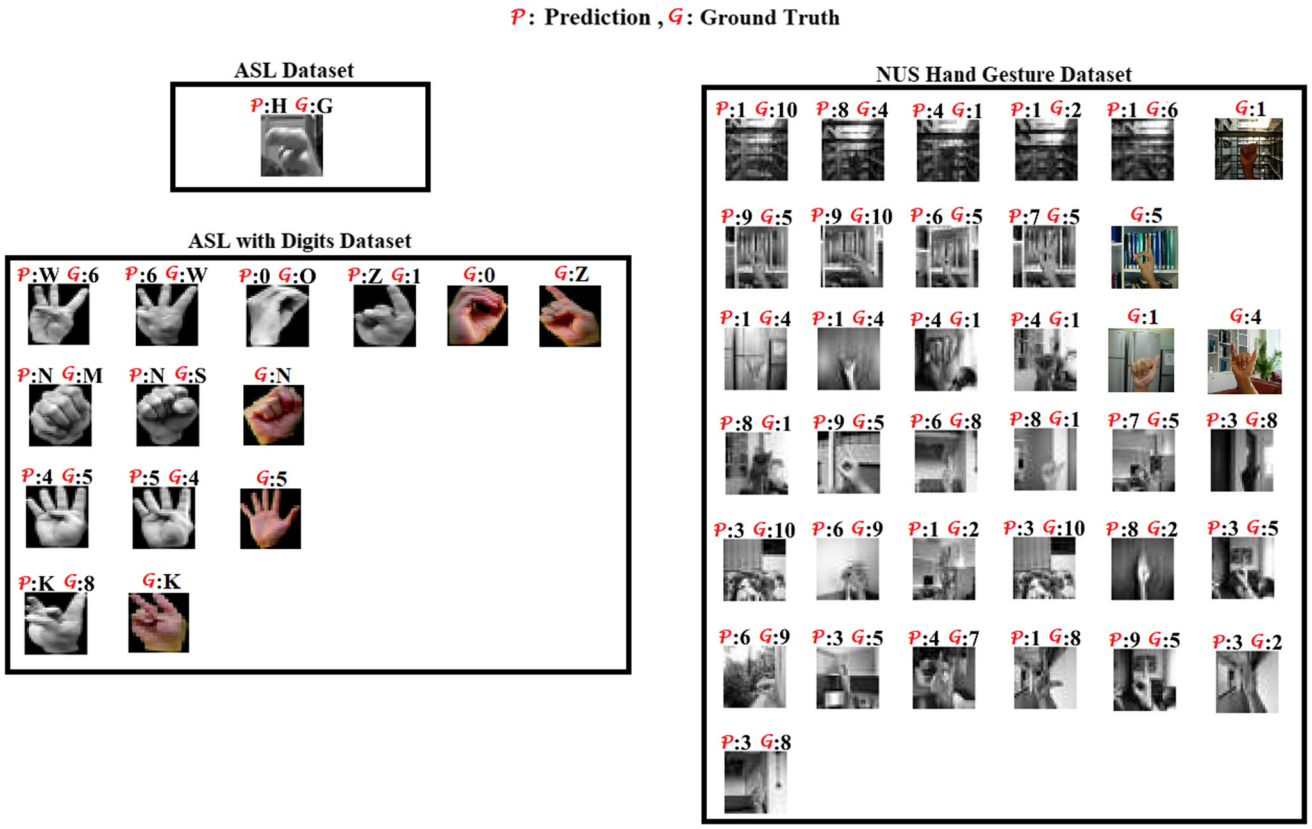

Fig. 13 One grayscale image from ASL dataset, nine grayscale images from ASL with digits dataset and 32 grayscale images from NUS hand gesture dataset which were misclassified; color images illustrated are for comparison purpose to provide clearer insight

the first and 512 neurons in the second, both with a dropout rate of 50%. The output from the first and second path's fully connected layers is concatenated and subsequently used to calculate the output.

In comparison with CNN baseline, ADCNN and DAG-CNN, the proposed CNN–SPP is equipped with four consecutive convolutional layers with no pooling layer in between, since having pooling layer too early in the network architecture leads to loss of relevant information. This theory holds true when comparing CNN–SPP with three consecutive convolutional layers and CNN–SPP with four consecutive convolutional layers from the input image, in which the latter achieves better results in all the three datasets. In short, having multiple consecutive convolutional layers allows our network to better learn and extract more relevant information through SPP from the input image and suppress irrelevant features. Instead of conventional pooling, SPP is used to extract features from the feature maps, which enriches the features being fed into the fully connected layer, and this helps the network to better distinguish among different gestures.

For comparison, we train CNN baseline, ADCNN and DAG-CNN using a learning rate of 0.001, and a batch size of 16 across all three datasets. On the other hand, the number of epoch is set to 50 for ASL dataset, and 200 for ASL with digits and NUS hand gesture datasets. Apart from that, softmax cross-entropy is used as the cost function with Adam optimizer. In the meantime, the batch normalization is applied to all convolutional and fully connected layers for all networks. We summarize our experimental results in Table 3 for comparison.

According to the experimental results in Table 3, ASL dataset gets comparable results among the networks tested. This is due to the sufficient amount of diverse images available for the networks to train on. Images in ASL with digits dataset pose a particular challenge with strong illumination changes presented in the images. The results obtained outperform all networks tested. NUS hand gesture dataset poses very tough challenges as it has numerous complex backgrounds and dissimilar illumination. CNN–SPP, however, still outperforms other networks in NUS hand gesture dataset. We portray the confusion matrices for all three datasets in Figs. 10, 11 and 12, respectively.

Figure 13 illustrates all grayscale images that were misclassified from ASL dataset, ASL with digits dataset and NUS hand gesture dataset, respectively. We discerned that four grayscale images in the first row of ASL with digits dataset were misclassified because of the same hand

gesture with no differentiation between the two classes of gesture, which are classes of (W, 6) and (0, O), while the only differentiation between classes of (Z, 1) is that Z is slightly tilted to the left. We also uncovered four mislabeled images which are supposed to be class number 4 instead of 5 in ASL with digits dataset, which are supposedly mislabeled by the creator itself. Out of the four mislabeled images, two images are misclassified.

In addition, we discovered two distinct complex backgrounds poses a problem for hand gesture recognition, as some gestures are hardly visible in these two backgrounds. The first and second rows of NUS hand gesture dataset in Fig. 13 depict nine grayscale images, from those two distinct complex backgrounds that were misclassified, and color images of those two complex backgrounds are provided for clearer illustration. We also discerned five misclassifications that overlap between gesture classes 1 and 4; the first image is shown in the first row, third image, remaining four images are in the third row. In general, some salient features of gestures in NUS hand gesture dataset are lost as they were downsized to 28×28. Possible solution to this particular problem is to perform data augmentation on the original sized images before downsizing to 28×28, and this, in turn, might help preserve salient features of gestures for network to differentiate.

5 Conclusion

In this paper, a spatial pyramid pooling (SPP)-integrated convolutional neural network (CNN), referred to as CNN–SPP, is outlined for vision-based hand gesture recognition. We discern that SPP alleviates the problem found in the conventional max-pooling operation and extends the features being fed into the fully connected layer which enables the network to better distinguish among different hand gestures. Apart from that, performing SPP on the earlier layer and feeding it directly to a fully connected layer help the gradients to flow more directly to the earlier layers, thus allowing the earlier layer to learn better. Our experimental results disclose that CNN–SPP outperforms the recent deep learning-driven instances evaluated on two American sign language (ASL) benchmarking datasets and one NUS hand gesture dataset.

Acknowledgements This research was supported by Fundamental Research Grant Scheme, Grant No. MMUE/190026.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Lim KM, Tan AW, Tan SC (2016) Block-based histogram of optical flow for isolated sign language recognition. J Vis Commun Image Represent 40:538–545
2. Lim KM, Tan AW, Tan SC (2016) A feature covariance matrix with serial particle filter for isolated sign language recognition. Expert Syst Appl 54:208–218
3. Lim KM, Tan AW, Tan SC (2017) A four dukkha state-space model for hand tracking. Neurocomputing 267:311–319
4. Kour KP, Mathew L (2017) Sign language recognition using image processing. Int J Adv Res Comput Sci Softw Eng 7(8):10
5. Kumar BP, Manjunatha M (2017) A hybrid gesture recognition method for American sign language. Indian J Sci Technol 10(1):1–12
6. He Y, Li G, Liao Y, Sun Y, Kong J, Jiang G, Jiang D, Tao B, Xu S, Liu H (2017) Gesture recognition based on an improved local sparse representation classification algorithm. Cluster Comput 22:10935–10946
7. Muthukumar K, Poorani S, Gobhinath S (2017) Vision based hand gesture recognition for Indian sign languages using local binary patterns with support vector machine classifier. Adv Natl Appl Sci 11(6):314–322
8. Hu Y (2018) Finger spelling recognition using depth information and support vector machine. Multimedia Tools Appl 77(21):29043–29057
9. Pariwat T, Seresangtakul, P (2017) Thai finger-spelling sign language recognition using global and local features with SVM. In: 2017 9th international conference on knowledge and smart technology (KST), pp 116–120. IEEE
10. Silanon K (2017) Thai finger-spelling recognition using a cascaded classifier based on histogram of orientation gradient features. Computational intelligence and neuroscience 2017
11. Jadooki S, Mohamad D, Saba T, Almazyad AS, Rehman A (2017) Fused features mining for depth-based hand gesture recognition to classify blind human communication. Neural Comput Appl 28(11):3285–3294
12. Zare AA, Zahiri SH (2018) Recognition of a real-time signer-independent static Farsi sign language based on fourier coefficients amplitude. Int J Mach Learn Cybernet 9(5):727–741
13. Nai W, Liu Y, Rempel D, Wang Y (2017) Fast hand posture classification using depth features extracted from random line segments. Pattern Recogn 65:1–10
14. Hu Y, Zhao HF, Wang ZG (2018) Sign language fingerspelling recognition using depth information and deep belief networks. Int J Pattern Recognit Artif Intell 32(06):1850018
15. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
16. Lim KM, Tan AWC, Lee CP, Tan SC (2019) Isolated sign language recognition using convolutional neural network hand modelling and hand energy image. Multimedia Tools Appl 78:1–28
17. Nakjai P, Katanyukul T (2019) Hand sign recognition for Thai finger spelling: an application of convolution neural network. J Signal Process Syst 91(2):131–146
18. Li Y, Wang X, Liu W, Feng B (2018) Deep attention network for joint hand gesture localization and recognition using static rgb-d images. Inf Sci 441:66–78
19. Kwolek B, Sako S (2017) Learning siamese features for finger spelling recognition. In: International conference on advanced concepts for intelligent vision systems, Springer, pp 225–236
20. Hosoe H, Sako S, Kwolek B (2017) Recognition of jsl finger spelling using convolutional neural networks. In: 2017 Fifteenth

IAPR international conference on machine vision applications (MVA), IEEE, pp 85–88
21. Gao Q, Liu J, Ju Z, Li Y, Zhang T, Zhang L (2017) Static hand gesture recognition with parallel cnns for space human-robot interaction. In: International conference on intelligent robotics and applications, Springer, pp 462–473
22. Kania, K, Markowska-Kaczmar U (2018) American sign language fingerspelling recognition using wide residual networks. In: International conference on artificial intelligence and soft computing, Springer, pp 97–107
23. Oliveira M, Chatbri H, Little S, Ferstl Y, O'Connor NE, Sutherland A (2017) Irish sign language recognition using principal component analysis and convolutional neural networks. In: 2017 international conference on digital image computing: techniques and applications (DICTA), IEEE, pp 1–8
24. Flores CJL, Cutipa AG, Enciso RL (2017) Application of convolutional neural networks for static hand gestures recognition under different invariant features. In: 2017 IEEE XXIV international conference on electronics, electrical engineering and computing (INTERCON), IEEE, pp 1–4
25. Alani AA, Cosma G, Taherkhani A, McGinnity T (2018) Hand gesture recognition using an adapted convolutional neural network with data augmentation. In: 2018 4th international conference on information management (ICIM), IEEE, pp 5–12
26. Arenas JOP, Moreno RJ, Beleño RDH (2018) Convolutional neural network with a dag architecture for control of a robotic arm by means of hand gestures. Contemp Eng Sci 11(12):547–557
27. Tazhigaliyeva N, Kalidolda N, Imashev A, Islam S, Aitpayev K, Parisi GI, Sandygulova A (2017) Cyrillic manual alphabet recognition in rgb and rgb-d data for sign language interpreting robotic system (slirs). In: 2017 IEEE international conference on robotics and automation (ICRA), IEEE, pp 4531–4536
28. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach F, Blei D (eds) Proceedings of the 32nd international conference on machine learning, Proceedings of machine learning research, vol 37, pp 448–456. PMLR, Lille, France
29. He K, Zhang X, Ren S, Sun J (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell 37(9):1904–1916
30. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
31. Pugeault N, Bowden R (2011) Spelling it out: real-time asl fingerspelling recognition. In: 2011 IEEE international conference on computer vision workshops (ICCV workshops), IEEE, pp 1114–1119
32. Barczak ALC, Reyes N.H, Abastillas M, Piccio A, Susnjak T (2011) A new 2d static hand gesture colour image dataset for asl gestures
33. Kumar PP, Vadakkepat P, Loh AP (2010) Hand posture and face recognition using a fuzzy-rough approach. Int J Humanoid Rob 7(03):331–356
34. Shin HC, Roth HR, Gao M, Lu L, Xu Z, Nogues I, Yao J, Mollura D, Summers RM (2016) Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. IEEE Trans Med Imaging 35(5):1285–1298
35. Zhang L, Yang F, Zhang YD, Zhu YJ (2016) Road crack detection using deep convolutional neural network. In: 2016 IEEE international conference on image processing (ICIP), IEEE, pp 3708–3712
36. Kagaya H, Aizawa K, Ogawa, M (2014) Food detection and recognition using convolutional neural network. In: Proceedings of the 22nd ACM international conference on Multimedia, pp. 1085–1088
37. Pigou L, Dieleman S, Kindermans PJ, Schrauwen B (2014) Sign language recognition using convolutional neural networks. In: European conference on computer vision, Springer, pp 572–578
38. Ma Y, Zhou G, Wang S, Zhao H, Jung W (2018) SignFi: sign language recognition using WiFi. Proc ACM Interact Mobile Wearable Ubiquit Technol 2(1):1–21