Sensors and Materials, Vol. 36, No. 2 (2024) 623–637
MYU Tokyo

S & M 3551

Human Hand Gesture Classification Based on Surface Electromyography Signals by Using a Vector–Kernel Convolutional Takagi–Sugeno–Kang Neuro-fuzzy Classifier

Cheng-Jian Lin,* Chun-Jung Lin, and Xin-Wei Lin

Department of Computer Science & Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan

(Received June 8, 2023; accepted January 12, 2024)

Keywords: fast Fourier transform (FFT), hand gesture classification, neuro-fuzzy network, surface electromyography (sEMG), vector–kernel convolution

Muscle–computer interfaces are devices that can identify the meaning of human bioelectrical signals, such as surface electromyography (sEMG) signals. sEMG signals can be obtained from arm-worn sensors and can be used to classify human gestures. In this paper, we propose a vector–kernel convolutional Takagi–Sugeno–Kang (TSK)-type neuro-fuzzy classifier (VK-CTNFC) to recognize human gestures represented by sEMG signals. First, vector–kernel convolution is used to extract the features of sEMG signals; this modification reduces the model parameters by half and increases the classification accuracy compared with those achieved with a conventional convolutional kernel. Second, the global average pooling method is used instead of the flattening method to improve feature fusion performance. Finally, a TSK-type neuro-fuzzy network is used for gesture classification. The publicly available dataset Ninapro DB1 was used in experiments for verifying the performance of the proposed VK-CTNFC. Data preprocessing was performed by wavelet denoising to smooth the sEMG waveform, and fast Fourier transform was used to convert time-domain sEMG signals into frequency-domain signals. Finally, the processed sEMG signals were input into the VK-CTNFC for training. The experimental results indicate that the proposed VK-CTNFC has an average accuracy of 87.18% and outperforms other reported methods.

1. Introduction

Human–computer interfaces (HCIs) are devices that enable users to control electronic devices interactively by using physical sensors, such as a mouse, keyboard, button, or touch surface. Although these physical sensors can provide efficient and accurate communication between the user and the device, the sensors must be within the user's reach. Therefore, they are inconvenient in some applications.

A muscle–computer interface (MCI) is a device that detects and decodes human muscle activity for interaction with a computer. Muscle signal sensors are worn on a human muscle and

*Corresponding author: e-mail: cjlin@ncut.edu.tw
https://doi.org/10.18494/SAM4681

ISSN 0914-4935

obtain signals when the muscle moves. The instructions represented by the signal are decoded and sent to the computer to achieve a task. Current applications of MCIs include muscle fatigue tracking,[1] electric wheelchair control,[2] vehicle-mounted device control solutions,[3] and sign language recognition.[4]

Approximately 541000 Americans were estimated to suffer from an upper extremity injury of any degree in 2005; this number is expected to double by 2025.[5] The main reasons for these injuries are trauma, followed by tumors or infectious diseases.[6] Patients with hand disabilities have reduced the ability to live independently; however, prosthetics can often improve their function. For example, patients with hand disabilities cannot use conventional physical HCI sensors, such as a mouse or buttons. Instead, patient muscle signals can be collected to identify an instruction to an electronic system. Prosthetic control is a commonly investigated application of MCI.[7–10] Surface electromyography (sEMG) signals are well suited for this application. Because an sEMG signal sensor is noninvasive (requiring only skin contact), such a sensor is suitable for long-term wearing.

Some researchers have used artificial intelligence methods to recognize human gestures represented in sEMG signals. Common artificial intelligence methods can be divided into conventional machine learning and deep learning methods. In machine learning, features in a dataset must be identified before producing a model. Phinyomark and Scheme[11] and Phinyomark *et al.*[12] used time- and frequency-domain features of EMG signals, respectively, as features of training data. Atzori *et al.*[13] used the K-nearest neighbor (KNN), support vector machine (SVM), random forest, and latent Dirichl*et all*ocation (LDA) classifiers to classify five EMG data features. Shenoy *et al.*[14] used linear SVM to classify eight gestures that enabled controlling a robotic arm with four degrees of freedom. Waris *et al.*[15] conducted machine learning experiments on ten able-bodied and six transradial amputees for seven consecutive days, and compared the classification results of LDA, artificial neural network (ANN), SVM, KNN, and decision trees; ANN had the highest classification accuracy. Conventional machine learning methods have some limitations; features must be manually selected for each problem by an expert with relevant professional knowledge. This process is tedious and might be insufficiently accurate; thus, testing different feature sets is challenging. Moreover, the feature classifier must be optimally selected for the corresponding classification task to obtain the best classification results.

Researchers have also used deep learning in MCI systems. Atzori *et al.*[16] used a modified LeNet model for prosthetic action classification. Wei *et al.*[17] used a multistream convolutional neural network (CNN) to divide sEMG input signals into multiple equal-sized signal streams; a CNN was trained on these signals to obtain features, and these features were fused to achieve gesture recognition. Olsson *et al.*[18] used the CNN topology to optimize the number of convolutional layers, kernel size, and the number of kernels. CNNs using simple architectures achieve similar accuracy to conventional machine learning classification methods.[16] Because deep learning methods can automatically extract key features in input data to improve classification accuracy, deep learning methods have been widely used in recent years.[19]

CNNs are effective not only for image classification problems[20–22] but also for extracting the spatial features of channels in sEMG signal maps. However, conventional CNNs have

numerous parameters, resulting in a high computational cost of model training and an unacceptable delay in obtaining results for real-time tasks. A vector–kernel CNN[23] was used to replace the convolution operation in a conventional CNN by reducing the number of parameters. In conventional CNNs, flattening is first conducted to expand the features before they are input to the classifier. Various effective feature fusion methods are often used to reduce the number of neurons.[24] Fully connected neural networks (FCNNs) are often used as classifiers in conventional CNNs. Because FCNNs are black boxes with numerous parameters, neuro-fuzzy networks have been proposed as a replacement for FCNNs.[25]

In this paper, we propose a vector–kernel convolutional Takagi–Sugeno–Kang (TSK)-type neuro-fuzzy classifier (VK-CTNFC) to classify human hand poses from sEMG signals. The main contributions of this study are as follows:

1. An effective VK-CTNFC model that combines vector–kernel convolution and a TSK-type neuro-fuzzy network (TNFN) is proposed for classifying hand poses from sEMG signals.
2. Global average pooling (GAP) is used to fuse data features in the fusion layer of the VK-CTNFC.
3. A TNFN is used to replace an FCNN in the traditional CNN architecture, which considerably reduces the number of training parameters. Compared with other models, the proposed VK-CTNFC has not only fewer parameters but also higher accuracy.

The remainder of this paper is organized as follows. Section 2 presents the system flowchart and the structure of the proposed VK-CTNFC. Section 3 describes the experimental results obtained for the proposed VK-CTNFC models. Finally, Sect. 4 provides the conclusions of this study and recommendations for future research.

2. Materials and Methods

The flowchart of the overall proposed system is depicted in Fig. 1. First, the sEMG signal of the gesture is preprocessed by wavelet denoising (WD) and fast Fourier transform (FFT). Second, the preprocessed signals are input into the proposed VK-CTNFC for gesture recognition and classification. Finally, the represented gesture is output.

2.1 Signal preprocessing

During sEMG signal collection, key features in the signals are unclear because of the effects of factors such as noise. WD and FFT, which are two data preprocessing techniques, are used to reduce the interference of these factors and translate the sEMG signal into the frequency domain, respectively. Figure 2 presents a flowchart of the signal preprocessing method.

1) Wavelet Denoising: WD[26] is based on wavelet transform. First, the time series of each channel of an sEMG signal is decomposed by continuous wavelet transform (CWT) to obtain wavelet coefficients. CWT, which is the product of the time series $x(t)$ and $\psi_{a,b}(t)$, is defined as

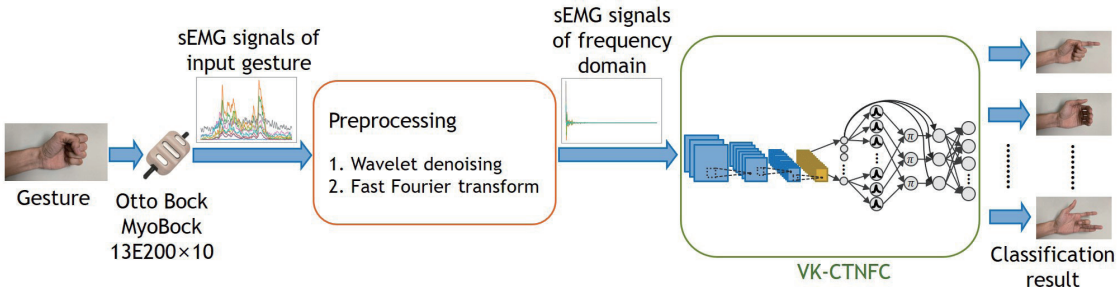


Fig. 1. (Color online) Flowchart of overall proposed system.

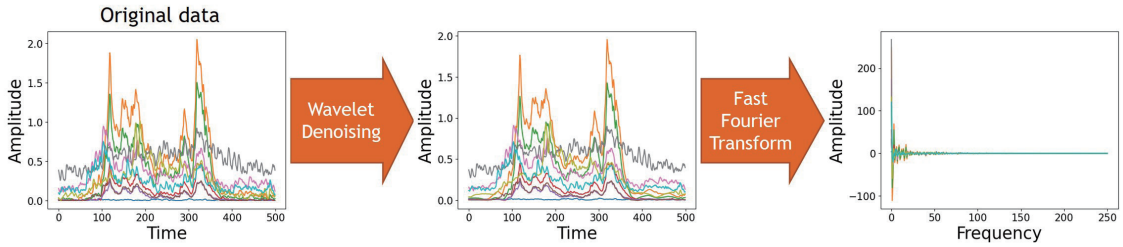


Fig. 2. (Color online) Signal preprocessing method.

$$W(a,b)=\int x(t)\frac{1}{\sqrt{a}}\psi_{a,b}(t)dt, \tag{1}$$

where $x(t)$ is the input signal and ψ is the mother wavelet, which contains a ($a \neq 0$) and b. The parameters a and b represent dilation and translation, respectively. The wavelet ψ is defined as

$$\psi_{a,b}(t)=\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right). \tag{2}$$

After the wavelet coefficient w is obtained, an appropriate threshold T is set as the wavelet coefficient. All wavelet coefficients below the threshold T are set as 0. A soft threshold function is used, and this function is defined as

$$soft(w,T)=\begin{cases}\operatorname{sgn}(w)(w-T), & \text{if } |w|>T\\ 0. & \text{otherwise}\end{cases} \tag{3}$$

Finally, the wavelet signal is reconstructed to obtain the noise-free signal.

2) Fast Fourier Transform: FFT is based on discrete Fourier transform (DFT) and characterized by its exploitation of the symmetric property of complex multiplication in a complex plane. In FFT, multiple multiplications with symmetric properties are combined into a single term to reduce computation. The structure of the original mathematical model is unchanged; thus, FFT achieves the same calculation result as DFT does but more rapidly. DFT is defined as

$$D(k)=\sum_{n=0}^{N-1} d(n)\exp\left[-2\pi i\frac{nk}{N}\right], \quad k=0,1,\ldots,N-1. \tag{4}$$

After WD, FFT is used to convert an sEMG signal represented in the time domain into a frequency-domain representation.

2.2 Proposed VK-CTNFC

The architecture of the proposed VK-CTNFC is shown in Fig. 3. This classifier contains an input layer, a feature extraction layer, a feature fusion layer, a fuzzification layer, a rule layer, a consequent layer, and an output layer. The goal of the VK-CTNFC is to classify an input sEMG signal as the intended gesture.

1) Input Layer: The input signal of the input layer is represented by I and expressed as

$$I=\begin{bmatrix} I_{0,0} & \cdots & I_{0,C} \\ \vdots & \ddots & \vdots \\ I_{F,0} & \cdots & I_{F,C} \end{bmatrix} \quad F=0,1,\ldots,499, \quad C=0,1,\ldots,9, \tag{5}$$

where F and C represent the length and the number of channels of the input signal, respectively.

2) Feature Extraction Layer: The feature extraction approach of the proposed method involves the use of the vector–kernel convolution operation to reduce the number of network parameters. Figure 4(a) illustrates the original $k \times k$ convolution operation. Figure 4(b) shows

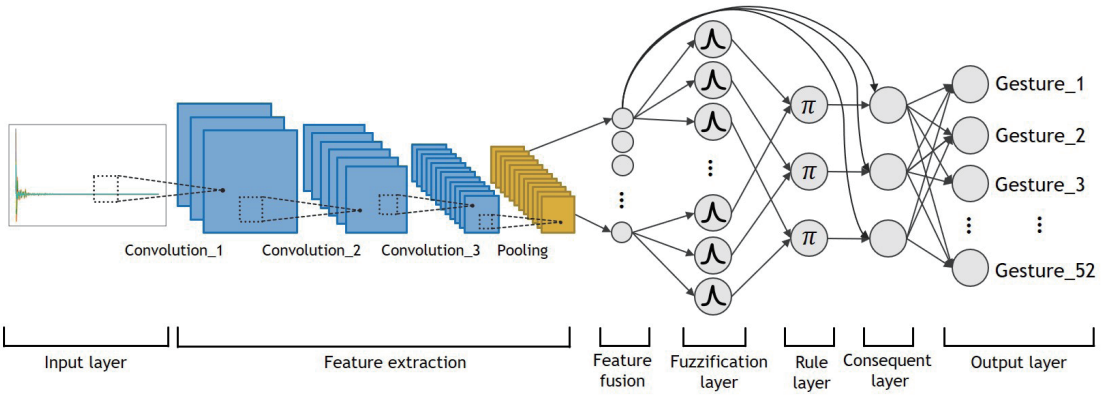
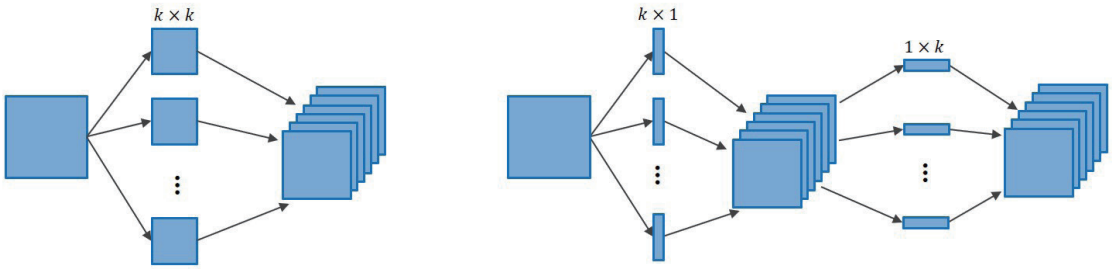


Fig. 3. (Color online) Architecture of proposed VK-CTNFC.



(a) (b)

Fig. 4. (Color online) (a) Original $k \times k$ convolution operation. (b) Two vector–kernel convolution operations ($k \times 1$ and $1 \times k$) equivalent to the original convolution operation.

the use of two vector–kernel convolution operations ($k \times 1$ and $1 \times k$) that are equivalent to the original convolution operation.

The vector–kernel convolution operation is defined as

$$y_{conv} = f_{act.}\left(\sum_{i=1}^{N} I_i * K_{ij} + B_j\right), \tag{6}$$

where I_i is the input matrix of the convolution, N is the number of feature maps, K_{ij} is the jth kernel matrix of the ith layer, and B_j is the bias. Finally, a nonlinear activation function is applied to each element to produce the output feature map matrix y_{conv}. The size n_o of the output feature map matrix y_{conv} is calculated as

$$n_o = \left\lfloor \frac{n_i + 2p - k}{s} \right\rfloor + 1, \tag{7}$$

where n_i is the size of the input matrix, p is the padding size, k is the kernel size, and s is the stride length.

3) Feature Fusion Layer: A commonly used operation in feature fusion layers is the global pooling operation. In the feature fusion layer of the proposed VK-CTNFC, GAP and global max pooling (GMP) are used to compress and reduce the dimension of the feature map, as shown in Fig. 5.

a) Global average pooling: GAP involves the averaging of each feature map output by the convolutional layer. In GAP, the output size is $1 \times f$. GAP is defined as

$$y_{fusion}[f] = y_{GAP}[f] = Avg\left(y_{conv}[f]\right). \tag{8}$$

b) Global max pooling: GMP involves the identification of the maximum value of each feature map output by the convolutional layer. In GMP, the output size is $1 \times f$. GMP is defined as

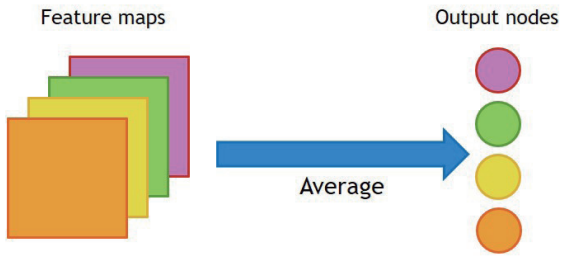


Fig. 5. (Color online) Global pooling process.

$$y_{fusion}[f] = y_{GMP}[f] = \mathrm{Max}\left(y_{conv}[f]\right). \tag{9}$$

4) Fuzzification Layer: Fuzzy rules are formulated as if–then statements and can be expressed as

$$\text{If } I_1 \text{ is } A_1^j \text{ and } I_2 \text{ is } A_2^j \ldots \text{and } I_i \text{ is } A_i^j \ldots \text{and } I_n \text{ is } A_n^j, \text{then } y = y^j(I), \tag{10}$$

where I_i represents the input, n represents the dimension of the input, A_i^j represents the fuzzy set of the antecedent part, j represents the jth rule, and $y^j(I)$ represents the fuzzy set of the consequent part. The Gaussian membership function is adopted for the fuzzy set of the antecedent. The firing strength of the membership function is defined as

$$M_i^j = \exp\left[-\left(\frac{I_i - m_i^j}{SD_i^j}\right)^2\right], \tag{11}$$

where m_i^j and SD_i^j are the mean and standard deviation, respectively.

a) Rule layer: The firing strength of the membership function is used to perform a fuzzy AND operation to obtain the firing strength of a fuzzy rule. The fuzzy AND operation adopted in this study involves using the product operation and is defined as

$$Rule_j = \prod_{i=1}^{n} M_i^j. \tag{12}$$

b) Consequent layer: The consequent part of the fuzzy law is expressed in TSK form and is a linear combination of the inputs. The expression is

$$y^j(I) = a_0^j + \sum_{i=1}^{n} a_i^j \cdot I_i^j, \tag{13}$$

where $a_0^j, a_1^j, a_2^j, \ldots, a_n^j$ are adjustable parameters.

c) Output layer: The output of the TNFN is defined as

$$y_{output} = \frac{1}{\sum_{j=1}^{r} M_i^j} \sum_{j=1}^{r} y^j (I) M_i^j. \tag{14}$$

3. Experimental Results

3.1 Dataset

The public dataset Ninapro DB1[18] was used for model training and testing. This dataset was collected using ten Otto Bock MyoBock 13E200 electrodes as sEMG signal sensors. A total of 27 healthy subjects (20 men and 7 women) performed 52 unique movements ten times each. Two subjects were left-handed, and the remaining subjects were right-handed. The heights of the subjects ranged from 155 to 185 cm, and their ages ranged from 22 to 40 years. The subjects were asked to mimic the hand movements shown to them in a movie played on a computer screen. They performed three exercises in the experiment: 1) basic movements of the fingers; 2) isometric and isotonic hand configurations and basic wrist movements; and 3) grasping and functional movements. Each action in the Ninapro DB1 dataset was continuously recorded for 5 s at a sampling frequency of 100 Hz; thus, the length of each data set was 500 sampling points.

3.2 Data preprocessing results

An original signal from Ninapro DB1 is shown in Fig. 6(a). First, WD was used to perform noise filtering. The denoised sEMG signal is displayed in Fig. 6(b). FFT was then used to convert the time-domain sEMG signal into the frequency domain, as depicted in Fig. 6(c).

In the dataset, the signals of each category were randomly divided into training, verification, and test sets with a ratio of 6:2:2. Table 1 shows the number of signals in each set.

3.3 Model evaluation metrics

To evaluate the performance of the proposed VK-CTNFC, four evaluation metrics, namely, accuracy, precision, recall, and *F*1-score, were used. These evaluation metrics are defined as

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \tag{15}$$

$$Precision = \frac{TP}{TP + FN}, \tag{16}$$

$$Recall = \frac{TP}{TP + FP}, \tag{17}$$

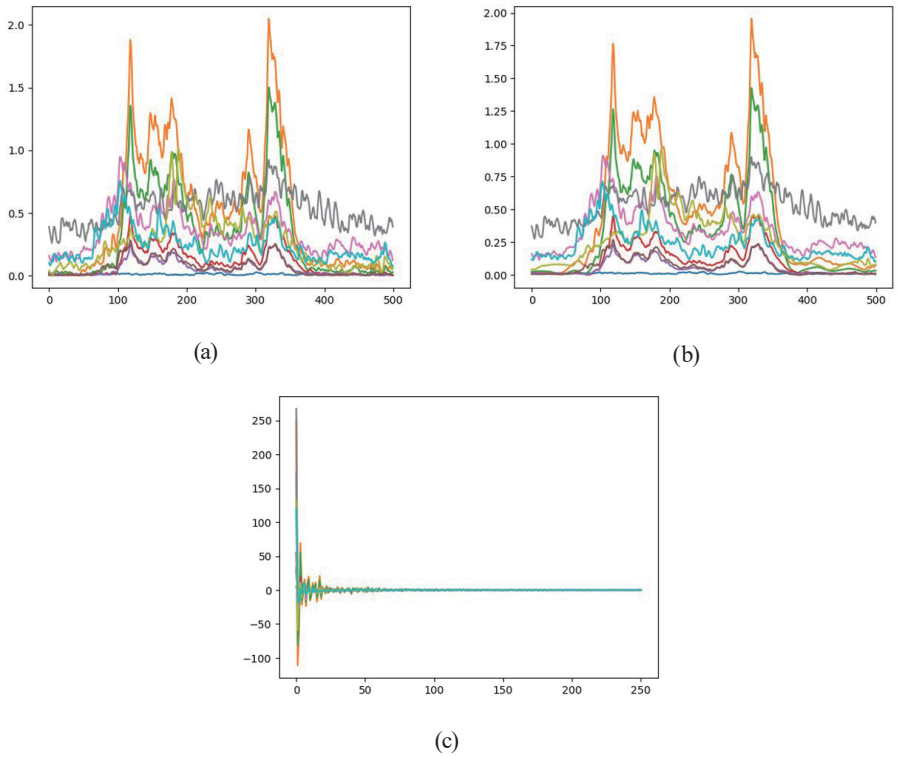


Fig. 6. (Color online) (a) Original, (b) denoised, and (c) FFT-transformed sEMG signals.

Table 1
Sizes of the training, validation, and test sets.

Dataset	Number of data
Training	8426
Validation	2808
Test	2808
Total	14042

$$F1-score=\frac{2\times Precision\times Recall}{recision+Recall}, \tag{18}$$

where *TP*, *TN*, *FP*, and *FN* indicate true positive, true negative, false positive, and false negative, respectively.

3.4 Performance analysis and comparison

VK-CTNFC architectures with various numbers of vector–kernel convolutional layers were analyzed to optimize the classification performance. The setting values of each convolutional layer are shown in Table 2. Table 3 presents the accuracy values of various vector–kernel convolution layers; the architecture with three vector–kernel convolutional layers achieved the highest classification accuracy.

Table 2
Settings for each convolutional layer.

Convolution layer	Number of filters	k	Activation
1	100	5	ReLU
2	200	5	ReLU

Table 3
Accuracy values achieved with different numbers of vector–kernel convolutional layers.

No. of vector–kernel convolution layers	Accuracy (%)
1	49.47
2	85.29
3	87.18
4	85.61

To ensure the accuracy of the evaluation, the proposed VK-CTNFC was tested three times, and the results were averaged (Table 4). Figure 7 presents the confusion matrix obtained for the VK-CTNFC.

A conventional CNN with a three-layer 5 × 5 kernel and VK-CTNFC models with three-layer 5 × 1 and 1 × 5 vector–kernel convolutions were trained, and their performance characteristics were compared. Various preprocessing techniques, namely, no preprocessing, WD, FFT, and WD + FFT, were used to investigate their effects on classification performance. The fusion methods of GAP and GMP were also compared. Table 5 presents the results obtained with the conventional CNN and VK-CTNFC model. The original CNN has 1.77 times the total number of training parameters of the VK-CTNFC models (2064400 and 1163000, respectively). For feature fusion, models using GAP had a higher accuracy than those using GMP because the results of GMP might be excessively affected by the maximum value of the feature map. The use of a single data preprocessing method, whether WD or FFT, improved the classification accuracy; however, the combination of WD and FFT with GAP feature fusion resulted in the highest classification accuracy (87.18%). The accuracy, precision, recall, and $F1$-score of the optimized model (VK-CTNFC with WD + FFT and GAP) were 87.18, 88.43, 86.68, and 87.53%, respectively.

3.5 Comparison of proposed method with existing methods

The accuracy of the proposed VK-CTNFC was compared with that of other classifiers proposed in the literature by using the same dataset. The methods selected for comparison comprised conventional machine learning methods, namely, the random forest method of Atzori *et al.*[13] and the LDA and least-squares-SVM (LS-SVM) method of Nazemi and Maleki[27], and deep learning methods, namely, the multistream CNN architecture of Wei *et al.*,[17] the CNN architecture of Atzori *et al.*,[16] and the CNN topologies method of Olsson *et al.*[18]

The deep learning methods had one of two input methods: the multistream CNN or single-stream CNN method [Figs. 8(a) and 8(b), respectively]. The multistream CNN method involves dividing the input sEMG signal into multiple equal-sized signals and inputting them into

Table 4
Evaluation indices for VK-CTNFC.

No. of experiment times	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)
1st	87.29	88.46	86.92	87.67
2nd	87.04	88.42	86.60	87.48
3rd	87.22	88.42	86.53	87.45
Average	87.18	88.43	86.68	87.53

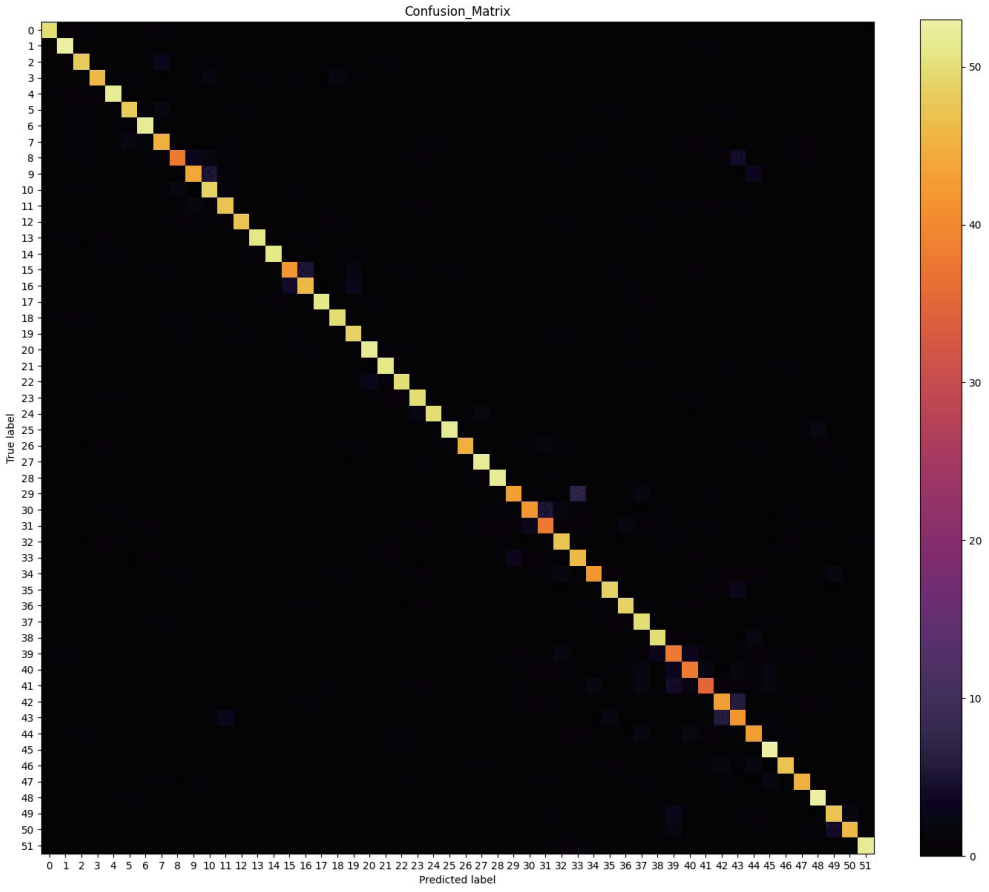
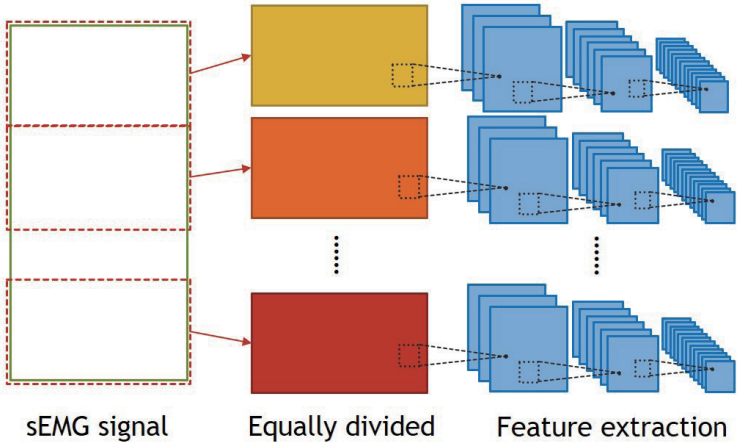


Fig. 7. (Color online) Confusion matrix for VK-CTNFC.

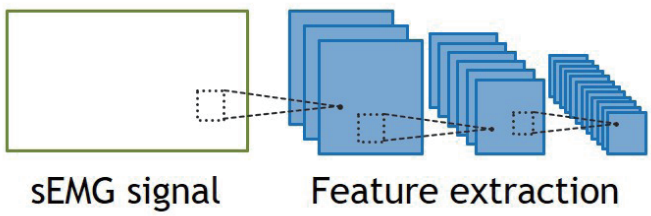
multiple independent CNN architectures for feature extraction and classification. By contrast, in the single-stream CNN method, sEMG signals are directly extracted and classified without segmentation. The proposed VK-CTNFC architecture uses a single-stream CNN approach. Table 6 shows a performance comparison of the aforementioned model architectures. The accuracy of the proposed VK-CTNFC architecture (87.1%) was higher than those of the other methods. In particular, its accuracy was 5.78% higher than those of other single-stream deep learning methods.[18] Thus, the results indicate that our architecture substantially outperforms similar architectures proposed in the literature.

Table 5
Performance of a CNN and two VK-CTNFC models with various preprocessing and pooling methods.

Models	Preprocessing	Fusion	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)	Total number of parameters
Original CNN	None	GAP	84.19	85.76	83.72	84.71	2064400
		GMP	82.48	84.35	80.88	82.55	
	WD	GAP	84.22	86.18	83.12	84.59	
		GMP	81.27	83.10	80.20	81.59	
	FFT	GAP	86.11	87.71	85.29	86.46	
		GMP	83.01	85.00	81.68	83.26	
	WD+FFT	GAP	86.15	87.34	85.87	86.43	
		GMP	83.75	85.29	83.29	84.25	
VK-CTNFC	None	GAP	86.15	87.87	85.14	86.45	1163000
		GMP	84.72	86.51	83.91	85.17	
	WD	GAP	86.86	87.93	86.25	87.06	
		GMP	85.26	86.39	84.53	85.43	
	FFT	GAP	86.61	88.00	85.97	86.96	
		GMP	86.18	87.70	85.19	86.41	
	WD+FFT	GAP	87.18	88.43	86.68	87.53	
		GMP	84.47	86.04	83.68	84.82	



(a)



(b)

Fig. 8. (Color online) (a) Multistream and (b) single-stream CNN methods.

Table 6
Accuracy comparison of various model architectures.

Architecture		Method	Accuracy (%)
RF by Atzori *et al.*[13]		Machine learning	75.32
LDA by Nazemi and Maleki[27]		Machine learning	84.23
LS-SVM by Nazemi and Maleki[27]		Machine learning	85.19
Multistream CNN by Wei *et al.*[17]		Multistream CNN	85.0
CNN by Atzori *et al.*[16]		Single-stream CNN	66.59 ± 6.40
CNN topologies by Olsson *et al.*[18]		Single-stream CNN	81.4 ± 4.0
Our method	VK-CTNFC with only WD	Single-stream CNN	86.86
	VK-CTNFC with only FFT	Single-stream CNN	86.61
	VK-CTNFC with WD and FFT	Single-stream CNN	87.18

Table 7
Number of trainable parameters for various architectures.

Architecture	Stream type	Trainable parameters
Multistream CNN by Wei *et al.*[17]	Multistream	8.69×10^6
Original CNN	Single-stream	2.06×10^6
VK-CTNFC	Single-stream	1.16×10^6

We also compared the numbers of trainable parameters of the proposed architecture and two other architectures (Table 7). The proposed VK-CTNFC architecture required considerably fewer trainable parameters than did the other two architectures (i.e., multistream and single-stream architectures).

4. Conclusions

In this paper, the VK-CTNFC, which combines vector–kernel convolution and the TNFN, is proposed to recognize human gestures from sEMG signals. In the VK-CTNFC, vector–kernel convolution is used to extract the features of sEMG signals; this modification reduces the number of parameters by half compared with that achieved with a conventional CNN. The GAP method is used for feature fusion, and a TNFN is then used for gesture classification. The publicly available dataset Ninapro DB1 was used to conduct experiments to verify the performance of the proposed VK-CTNFC. In our experiments, WD and FFT were used for data preprocessing. The experimental results indicate that the accuracy, precision, recall, and $F1$-score of the VK-CTNFC are 87.18, 88.43, 86.68, and 87.53%, respectively, and this classifier outperforms similar models proposed in previous relevant studies. Moreover, the model only requires 1163000 parameters.

Because many parameters of the proposed VK-CTNFC model must be determined by trial and error, developing the model is time-consuming; therefore, this model must be optimized further. In future research, we intend to use the Taguchi method[28] or uniform experimental design method[29] to obtain the optimal parameters of the proposed model by using a small number of experiments.

References

1 Z. Li, M. Hayashibe, C. Fattal, and D. Guiraud: IEEE Comput. Intell. Mag. **9** (2014) 38. https://doi.org/10.1109/MCI.2014.2307224
2 G. Jang and Y. Choi: Proc. 2014 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IEEE, 2014) 3549–3554
3 R. N. Khushaba, S. Kodagoda, D. Liu, and G. Dissanayake: Comput. Methods Programs Biomed. **110** (2013) 137. https://doi.org/10.1016/j.cmpb.2012.11.002
4 V. E. Kosmidou and L. J. Hadjileontiadis: IEEE Trans. Biomed. Eng. **56** (2009) 2879. https://doi.org/10.1109/TBME.2009.2013200
5 K. Ziegler-Graham, E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and R. Brookmeyer: Arch. Phys. Med. Rehabil. **89** (2008) 422. https://doi.org/10.1016/j.apmr.2007.11.005
6 W. R. Frontera and J. K. Silver: Fondamenti di Medicina Fisica e Riabilitativa (Rome, 2004) (in Italian).
7 A. Calado, F. Soares, and D. Matos: Proc. 2019 IEEE Int. Conf. Autonomous Robot Systems and Competitions (IEEE, 2019) 1–6. https://doi.org/10.1109/ICARSC.2019.8733629
8 W. Guo, P. Yao, X. Sheng, H. Liu, and X. Zhu: Proc. 2014 IEEE Int. Conf. Systems, Man, and Cybernetics (IEEE, 2014) 2192–2197. https://doi.org/10.1109/SMC.2014.6974249
9 C. Choi, B. Rim, and J. Kim: Proc. 2011 IEEE Int. Conf. Rehabilitation Robotics (IEEE, 2011) 1–5. https://doi.org/10.1109/ICORR.2011.5975386
10 S. Ida-Maria, K. R. Lyons, A. Nemchuk, S. D. Muroff, and S. S. Joshi: Hum. Mov. Sci. **47** (2016) 60. https://doi.org/10.1016/j.humov.2015.12.003
11 A. Phinyomark and E. Scheme: Proc. 2018 40th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (IEEE, 2018) 5236–5240. https://doi.org/10.1109/EMBC.2018.8513427
12 A. Phinyomark, P. Phukpattaranont, and C. Limsakul: Expert Syst. Appl. **39** (2012) 7420. https://doi.org/10.1016/j.eswa.2012.01.102
13 M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A. G. M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, and H. Müller: Sci. Data **1** (2014) 140053. https://doi.org/10.1038/sdata.2014.53
14 P. Shenoy, K. J. Miller, B. Crawford, and R. N. Rao: IEEE Trans. Biomed. Eng. **55** (2008) 1128. https://doi.org/10.1109/TBME.2007.909536
15 A. Waris, I. K. Niazi, M. Jamil, K. Englehart, W. Jensen, and E. N. Kamavuako: IEEE J. Biomed. Health Informat. **23** (2019) 1526. https://doi.org/10.1109/JBHI.2018.2864335
16 M. Atzori, M. Cognolato, and H. Müller: Frontiers Neurorobot. **10** (2016) 9. https://doi.org/10.3389/fnbot.2016.00009
17 W. T. Wei, Y. Wong, Y. Du, Y. Hu, M. Kankanhalli, and W. D. Geng: Pattern Recogn. Lett. **119** (2019) 131. https://doi.org/10.1016/j.patrec.2017.12.005
18 A. E. Olsson, A. Björkman, and C. Antfolk: Comput. Biol. Med. **120** (2020) 103723. https://doi.org/10.1016/j.compbiomed.2020.103723
19 Y. LeCun, Y. Bengio, and G. Hinton: Nature **521** (2015) 436. https://doi.org/10.1038/nature14539
20 A. Krizhevsky, I. Sutskever, and G. E. Hinton: Commun. ACM **60** (2017) 84. https://doi.org/10.1145/3065386
21 A. El-Sawy, E. B. Hazem, and M. Loey: 2016 Proc. Int. Conf. Advanced Intelligent Systems and Informatics (2016) 566–575. https://doi.org/10.1007/978-3-319-48308-5_54
22 F. Sultana, A. Sufian, and P. Dutta: Proc. 2018 4th Int. Conf. Research in Computational Intelligence and Communication Networks (IEEE, 2018) 122–129. https://doi.org/10.1109/ICRCICN.2018.8718718
23 J. Ou and Y. Li: Neurocomputing **330** (2019) 253. https://doi.org/10.1016/j.neucom.2018.11.028
24 M. Lin, Q. Chen, and S. Yan: Proc. Int. Conf. Learning Representation (2013) 1–10. https://doi.org/10.48550/arXiv.1312.4400
25 M. A. Velez, O. Sanchez, S. Romero, and J.M. Andujar: Appl. Soft Comput. **10** (2010) 578. https://doi.org/10.1016/j.asoc.2009.08.027
26 C. F. Jiang, and S. L. Kuo: Proc. 2007 29th Annual Int. Conf. IEEE Engineering in Medicine and Biology Society (IEEE, 2007) 1868–1871. https://doi.org/10.1109/IEMBS.2007.4352679
27 A. Nazemi and A. Maleki: Proc. 2014 4th Int. Conf. Computer and Knowledge Engineering (IEEE, 2014) 18–22. https://doi.org/10.1109/ICCKE.2014.6993343
28 C. J. Lin, X. Y. Lin, and J. Y. Jhang: Sens. Mater. **34** (2022) 3569. https://doi.org/10.18494/SAM4044
29 C. J. Lin, and S. Y. Jeng: Diagnostics **10** (2020) 662. https://doi.org/10.3390/diagnostics10090662

About the Authors

Cheng-Jian Lin received his B.S. degree in electrical engineering from Ta Tung Institute of Technology, Taipei, Taiwan, R.O.C., in 1986 and his M.S. and Ph.D. degrees in electrical and control engineering from National Chiao Tung University, Taiwan, R.O.C., in 1991 and 1996, respectively. Currently, he is a chair professor of the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan, R.O.C. His current research interests are in machine learning, pattern recognition, intelligent control, image processing, intelligent manufacturing, and evolutionary robots. (cjlin@ncut.edu.tw)

Chun-Jung Lin received his MS degree from China Medical University, Taichung, Taiwan, R.O.C., in 2001. He received his Ph.D. degree from the Department of Management of Information System, National Chung Cheng University, Chia-Yi, Taiwan, R.O.C., in 2012. Currently, he is an assistant professor in the Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan, R.O.C. His research interests include database systems, AI applications, healthcare information management, and software engineering. (phdraymond@ncut.edu.tw)

Xin-Wei Lin received his B.S. degree from the Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung, Taiwan, R.O.C., in 2022. Currently, he is a graduate student of the Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung, Taiwan. His current research interests include deep learning, neural fuzzy systems, and computer vision and applications. (swsteven@kimo.com)