TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 2508 – 2525
© TÜBİTAK
doi:10.55730/1300-0632.3952

Utilizing motion and spatial features for sign language gesture recognition using cascaded CNN and LSTM models

Hamzah LUQMAN*, **El-Sayed M. El-ALFY**
Department of Information and Computer Science, College of Computing and Mathematics,
Interdisciplinary Research Center for Intelligent Secure Systems (IRC-ISS),
SDAIA-KFUPM Joint Research Center for Artificial Intelligence,
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

Received: 14.05.2021 • **Accepted/Published Online:** 20.08.2022 • **Final Version:** 28.11.2022

Abstract: Sign language is a language produced by body parts gestures and facial expressions. The aim of an automatic sign language recognition system is to assign meaning to each sign gesture. Recently, several computer vision systems have been proposed for sign language recognition using a variety of recognition techniques, sign languages, and gesture modalities. However, one of the challenging problems involves image preprocessing, segmentation, extraction and tracking of relevant static and dynamic features related to manual and nonmanual gestures from different images in sequence. In this paper, we studied the efficiency, scalability, and computation time of three cascaded architectures of convolutional neural network (CNN) and long short-term memory (LSTM) for the recognition of dynamic sign language gestures. The spatial features of dynamic signs are captured using CNN and fed into a multilayer stacked LSTM for temporal information learning. To track the motion in video frames, the absolute temporal differences between consecutive frames are computed and fed into the recognition system. Several experiments have been conducted on three benchmarking datasets of two sign languages to evaluate the proposed models. We also compared the proposed models with other techniques. The attained results show that our models capture better spatio-temporal features pertaining to the recognition of various sign language gestures and consistently outperform other techniques with over 99% accuracy.

Key words: Sign language recognition, gesture recognition, sign language translation, action recognition, Arabic sign language recognition, CNN-LSTM

1. Introduction

Sign language is the main communication language of hearing impaired people. Recently, the World Health Organization (WHO) declared that more than 450 million people worldwide have a disabling hearing loss which forms about 5% of the world's population[1]. Disabling hearing loss refers to hearing loss in the better hearing ear greater than 40 decibels (dB) in adults and greater than 30 dB in children. This hard-hearing community depends primarily on the sign language as a main communication medium. Sign language is geographically specific; meaning there is no universal sign language but several languages exist around the world with regional variations such as the Arabic sign language (ArSL), Chinese sign language (CSL), and American sign language (ASL) [1]. Moreover, some countries may have a common spoken language but different sign languages such

*Correspondence: hluqman@kfupm.edu.sa

[1]World Health Organization (2021). Deafness and hearing loss [online]. Website https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss [accessed 22 February 2022].

as the British sign language (BSL) and ASL with completely different signs for alphabets and digits [2]. The differences between sign languages are mainly on the gesture meaning and the grammatical structure of sign sentences.

A sign language adopts manual and nonmanual gestures that are simultaneously employed to convey the meaning. Manual gestures involve hand movements whereas nonmanual gestures involve facial expressions, mouth shapes, and body postures [3]. Hand gestures have been the focus of several research as the primary component of the sign language utilized by individuals to express their thoughts [4]. However, they are typically accompanied with other gestures such as head movement and facial expressions to complement manual gestures. Facial expressions have several functions in sign language, e.g., convey emotion and feelings and clarify meaning or the degree of intense as well as the syntactical and grammatical information in sign language sentences. Sign language gestures can be static or dynamic depending on the motion usage. Static gestures keep the hand position fixed during signing and depend on hands and fingers shape and orientation [5]. This type of gestures is usually used in sign language for digits and alphabets and can be captured by spatial features. In contrast, dynamic gestures depend largely on the hands' motion and are most common in signing words/phrases of a sign language [6].

In the past two decades, a number of techniques have been proposed for automatic recognition of sign gestures. These techniques can be classified based on the recognized signs into static-based and dynamic-based recognition systems. The former category targets static signs that do not include motion such as digits and alphabets. For example, a convolutional neural network (CNN) model was used by Bheda and Radpour [7] to recognize ASL alphabets with an accuracy of 82.5% reported on a dataset consisting of five signers. The same dataset was used by Ranga et al. [8] and reported an accuracy of 97.01% using a CNN model with a different architecture. Another dataset consisting of 20 alphabets performed by 15 signers was used by Munib et al. [9] and an accuracy of 92.3% was reported using Hough descriptors and neural network classifiers. Another technique to classify ASL alphabets based on detecting important parts in the image was proposed by Zamani and Kanan [10]. These parts were fed into principal component analysis (PCA) and a neural network is used for classification yielding an accuracy of 99.88%. PCA was also used by Pan et al. [11] using Hu moments and support vector machine (SVM) achieving 94% and 99.8% accuracy on ASL and CSL datasets, respectively. Other researchers used Zernike moments (ZMs) for recognizing sign language alphabets with better results than Hu Moments, PCA, and Fourier descriptors [12–14].

In contrary to static-based systems, dynamic-based sign language recognition systems target motion to sign words, which provides a richer source of information. Most of the early techniques that have been proposed for dynamic signs depend on extracting statistical and geometric features of signs [15–27] while some of the recent techniques employed deep learning techniques for signs recognition [28–31]. Ameida et al. [16] reported an accuracy above 80% on a dataset consisting of 34 signs of the Brazilian sign language by extracting seven features from depth images to represent the structural elements of the signs. Zaki et al. [22] presented a novel method for feature extraction of dynamic sign language based on PCA and kurtosis position with motion chain code (MCC) to represent the shape and orientation of ASL signs to be classified using hidden Markov model (HMM). Another approach was proposed by Jin et al. [26] based on speeded up robust features (SURF) descriptor from 16 signs of ASL and SVM classifier giving an accuracy of 97.13%. Other classical features extraction techniques have been used for signs recognition such as dynamic time warping (DTW) [19, 24, 25], histogram of gradients (HOG) [32], wavelet and combined orientation histogram [33], and hands trajectory [34].

Liu et al. [35] fed the skeleton joint points of the signer hands captured using Kinect sensor to an LSTM model consisting of seven layers. This work was evaluated using two datasets consisting of 25K and 125K images and accuracies of 86% and 64% were achieved, respectively. Huang et al. [29] concatenated skeleton joint points with color and depth information to form an input consisting of three channels to a CNN model. This technique was evaluated on a dataset of 25 signs and an accuracy of 94.2% was reported. Integrating different information channels was also followed by Barros et al. [31]. Sabyrov et al. [36] used logistic regression for Kazakh-Russian sign language recognition. OpenPose was used to extract key points from manual gestures and facial expressions. The reported results showed that combining manual key points with mouth key points improved the accuracy by 7% whereas eyebrows key points improved the accuracy by only 0.5%. This conclusion was also reported by Elons et al. [37] who found that combining facial features with manual gestures improved the accuracy from 88% to 98%. This improvement in the accuracy is confirmed in [38]. Discrete cosine transform (DCT) was used by Rao and Kishore [39] for Indian sign language recognition. The 2D-DCT was used to extract features from signer head and hand which were detected using Sobel edge detector. This approach was evaluated on a dataset consisting of 18 signs and an accuracy of 90.58% was reported. In [40], DCT with HMM was used for ArSL recognition. A dataset consisting of 30 signs performed by 18 signers was used to evaluate this approach and accuracies of 96.74% and 94.2% were reported for signer dependent and independent modes, respectively. HMM was also used by Kelly et al. [41] to classify a set of statistical features extracted from the signer's hands and head.

An active appearance was used by Agris et al. [42] to detect signer's mouth and eyes and a numerical description was computed from those components. For signer's hands, a set of geometric features was extracted and concatenated with facial expression features. This fusion of features improved the accuracy of Danish sign language (DSL) recognition by around 1.5%. Sarkar et al. [43] reported an improvement by around 4.0% using 39 signs of ASL through combining manual and nonmanual gestures. SVM was used by Quesada et al. [44] for classifying manual and nonmanual markers captured using a natural user interface device, which captures hand shapes, body position, and face expressions using 3D cameras. This approach achieved an accuracy of 91.25% using 5 face gestures and 4 hand shapes of ASL. Kumar et al. [45] used two sensors to capture signers' hands and facial expressions of Irish sign language (ISL). They used Leap Motion Controller for manual gestures acquisition whereas Kinect was used for facial expression capturing. Then, HMM was used to recognize each component separately to be combined later using a Bayesian classification method. This combination boosted the recognition accuracy over single modality by 1.04%.

Though a number of approaches have been proposed for sign language recognition, there are several issues still open for research to integrate highly semantic representations of manual and nonmanual features [46]. In this work, we focus on video-based sign language recognition, propose three cascaded deep-learning models and evaluate their effectiveness, scalability, and computation time. Each model is composed of two components to extract spatial-temporal global-local features representations of various body parts postures and facial expressions. We employed CNN to extract spatial features from the sign gesture and fed them into a temporal feature learning technique. We intensively evaluated the performance of integrating vanilla and stacked LSTMs with CNN. In addition, we adopt an absolute temporal difference technique to track the motion in the video stream by preserving the differences between consecutive frames. This technique has been evaluated and compared with the other techniques using three datasets of two different sign languages.

The remaining sections of this article are as follows: The proposed methodology is explained in Section 2. Experimental results and analysis of the proposed models are discussed in Section 3. Conclusions and future works are highlighted in Section 4.

2. Methodology

In this study, three models have been proposed for video-based dynamic sign language recognition by integrating convolutional neural networks (CNN) with long short-term memory (LSTM) deep learning methods. To have a baseline, we used a vanilla LSTM with raw colored frames of sign gestures. Although LSTM is efficient for time-series data learning, it lacks the ability to learn the spatial correlations [47] that are important for recognizing sign language gestures. To address this limitation, we combined CNN with LSTM in three different schemes. The proposed method does not only consider body gestures but also facial expressions that provide valuable emotional information accompanying certain words/phrases. Three constructed models are evaluated using two input representations: raw colored frames and residue frames. The details of input representations and models are explained in the following subsections.

2.1. Input representations

Two representations of the captured sign gestures have been considered. First, each sign gesture is fed into the proposed models as a sequence of raw colored frames extracted from the sign video. However, the duration of each sign and consequently the number of frames are not uniform and vary from sign to another. Moreover, the duration of the same sign can differ from signer to another based on the speed of signing. To address this issue, we represented each sign by a fixed number of frames, N_S, which is sampled based on the average number of frames per database. For signs with frames less than N_S, we repeated the last frame whereas signs with frames more than N_S, we sampled frames at equal interval δ that is computed as follows:

$$\delta = \left\lceil \frac{N_F}{N_S} \right\rceil, \tag{1}$$

where N_F represents the total number of frames in the given video stream.

Dynamic sign gestures depend mainly on the motion. To preserve the signer's parts that change during signing and exclude the static parts of the gesture video, we used the absolute frame differences (AFDs). The use of absolute frame differences is very common in the literature of video analysis to detect changes and track objects from the pixel-based difference between consecutive frames [48]. Bigger changes between two video frames result in higher number of pixels' values being different. To demonstrate how AFD frames are computed, let F_t be the t^{th} frame in the video frames sequence and F_{t-1} be the preceding frame in the same sequence. The absolute differential frame ΔF_t is computed as follows:

$$\Delta F_t = |F_t - F_{t-1}| \ . \tag{2}$$

Figure 1 shows an illustrative example of consecutive frames and the resulting AFD frames. As shown in the figure, this technique preserves only the body parts that changed during the motion. In addition, it gets rid of static components, such as signer background and signer's static parts of the body.

2.2. Baseline model

An LSTM network is a type of recurrent neural network that exhibits temporal dynamic behavior in a time-series data. Standard recurrent neural network fails in learning time series data more than 10 discrete time steps

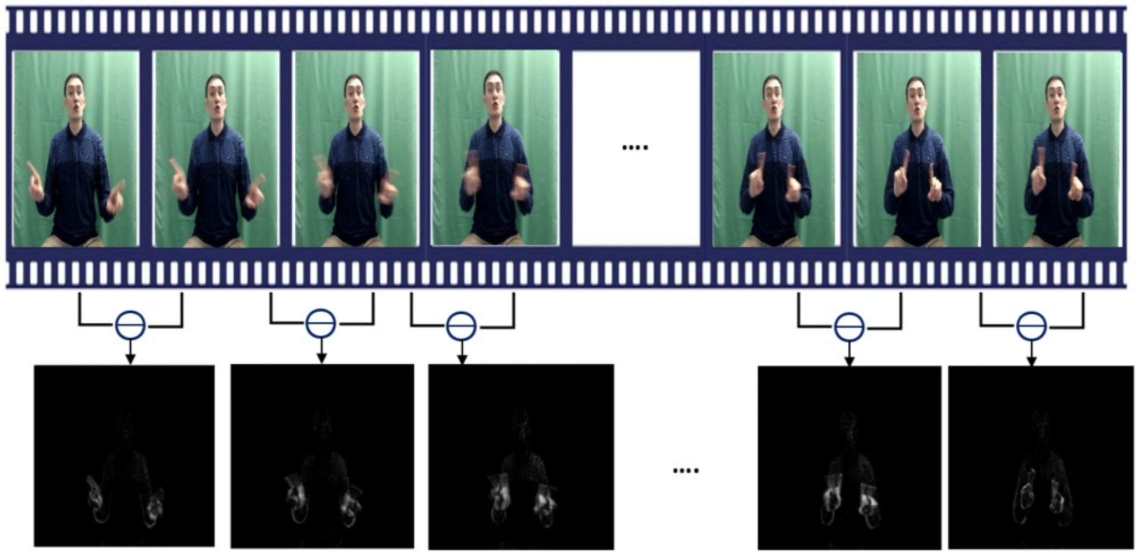

Figure 1. Absolute frame differences extracted from raw colored frames.

due to the vanishing gradient problem [49]. This results in limiting the range of contextual information that can be learnt [50]. LSTM addressed this issue by introducing input, forget and output gates, which allow for a better control over the gradient flow and enable better preservation of "long-range dependencies". This type of recurrent neural network has been introduced and applied with remarkable success to sequence prediction problems [51].

In this work, we used LSTM as a baseline model consisting of 1024 neurons selected empirically. Similar to other work in the literature, e.g., [52, 53], we tried several number of neurons (128, 512, 1024, and 2048) and we selected the number of neurons based on the best performance. This layer is followed by a dropout layer with 0.7 ratio to prevent model overfitting. The output of this layer is fed into a fully connected layer consisting of 512 neurons. A rectified linear (ReLU) activation function is used with this layer to handle the nonlinearity by setting the negative input to this function to zero. ReLU is computationally powerful and reduces the possibility of gradient vanishing [54]. This layer is followed by another dropout layer and the last layer consists of a softmax classifier with a number of neurons equivalent to the number of signs. To train this model, we used an Adam optimizer with a learning rate of 10^{-5} and a decay of 10^{-6}. We also used a crossentropy loss function to compute the loss value during model training.

2.3. Combining CNN with LSTM architectures

In this section, we describe three customized architectures for sign language recognition based on the integration of two components utilizing layers from CNN and LSTM. The proposed architectures, as explained in the following subsections, enable CNN to extract pertaining spatial features from input images whereas LSTM learns temporal features in the sequence of frames.

2.3.1. Model#1: CNN-LSTM model

This model consists of CNN component for extracting spatial features from video frames and an LSTM component to learn the relationship between sign frames. Each image is fed into a CNN consisting of six layers and the output of this model is fed into an LSTM consisting of 256 neurons. These layers are followed by a dropout layer and the output is classified using a softmax function as shown in Figure 2. This model has over 11.5 million trainable parameters. The model layers can be described as follows:

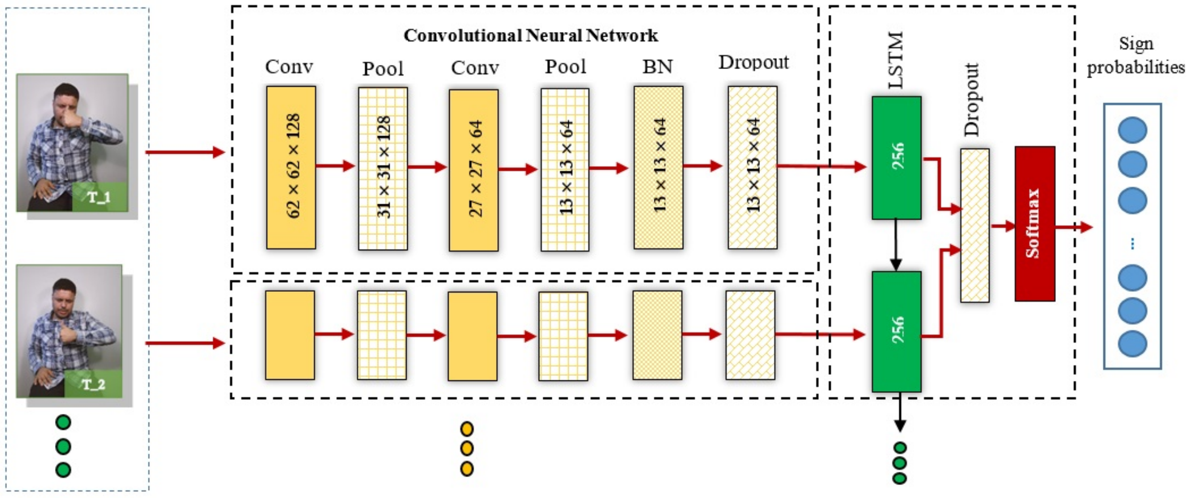

Figure 2. Proposed architecture showing various layers of model#1 (CNN-LSTM).

- *Convolution layers*: Each convolution layer applies a set of filters or kernels by convolution to extract features from input images. In our architecture, we used 128 filters of size 5×5 with a stride of two pixels horizontally and vertically with no padding. The number of filters and the size of each filter were selected empirically. The first convolution layer resulted in a features map of size $62 \times 62 \times 128$. All the convolution layers in this model are followed by a ReLU activation function that zeros negative input values. ReLU has been widely used in deep neural networks due to its ability in overcoming the gradient vanishing problem and its computation performance [54].

- *Pool layer*: The features map resulting from the convolution layer is downsampled using a max-pooling layer. The pooling layer helps in reducing the computational cost by reducing the number of learnable model parameters. In addition, it provides basic translation invariance to the changes in the position of the objects in the image. In this work, we used a max-pooling layer of size 2×2 with a stride of two pixels in both directions.

- *Batch normalization*: One of the critical issues associated with deep networks is the internal covariate shift problem that is a result of change in the network activation distribution due to the network parameters change during the training [55]. This problem can slow down the training and makes it notoriously difficult to train saturated nonlinearity models [55]. Batch normalization (BN) is a training technique to mitigate internal covariate shift for neural networks. This technique helps in standardizing the inputs to this layer for each minibatch and consequently reducing the number of training epochs required to train the model [56].

- *Dropout*: Deep networks usually suffer from overfitting the training dataset. To address this issue, we used a dropout technique that randomly drops out input units during training by setting them to zero [57]. This technique is a computationally effective regularization that improves the model generalization. We used a dropout with a rate of 0.7.

- *LSTM*: We used CNN model to extract features from each video frame separately. These features are fed into LSTM to learn the temporal features across video frames. More information about LSTM model can be found in Section 2.2. We used LSTM model with 256 neurons followed by a dropout layer. The output of the dropout layer is fed into the classification layer.

- *Softmax layer*: The output of the previous layers is fed into a softmax layer for classification. This layer assigns a probability value to each sign/class in the used dataset to be used for the classification. The number of neurons in this layer equals the number of signs in the used dataset.

- *Loss function and optimization*: The adjustment of neural networks parameters during the training process depends mainly on computing the loss value and optimizing the model parameters to minimize that loss. The loss function is used to quantify the goodness of the model using the predicted score with true labels. We used crossentropy loss that increases as the predicted probability diverges from the true classes. This function is defined for an input frame image x with its class label y as follows:

  $$\mathfrak{L}(x,y) = \delta(\overrightarrow{\phi(f_\theta(x))}, \overrightarrow{\varphi(y)}) \ , \tag{3}$$

  where $f_\theta(x)$ is the model function with learned parameter θ, $\phi(z_k) = e^{z_k}/\sum_i e^{z_i}$ is the prediction probability of outcome z_k for $k = 1...K$, K is the number of classes, $\overrightarrow{\varphi(y)} = [0,0,\ldots,1,\ldots,0,0]^T$ is a one-hot encoding column vector, and $\delta(\overrightarrow{a}, \overrightarrow{b}) = -\overrightarrow{b}^T \log(\overrightarrow{a})$.

  After computing the loss value, the model optimizes its parameters to minimize that loss. We used adaptive moment estimation (Adam) optimizer that can update the network weights iteratively based on the training data [58].

2.3.2. Model#2: CNN-SLSTM model

The previous CNN-LSTM model is composed of a single LSTM network followed by a dropout layer and a standard feed-forward output layer. This single LSTM network cannot usually learn complex patterns effectively. Therefore, we constructed a stacked LSTM on the top of the CNN block to learn higher levels of representations from the CNN extracted features. The gained performance improvement using layered architectures has also been reported with different problems [59].

In the proposed stacked model (model#2), there are two stacked LSTM layers. The first layer consists of 512 neurons while the second layer consists of 256 neurons. These two LSTM layers are stacked vertically and the output of the top LSTM layer is fed into a dropout layer to mitigate the possibility of overfitting. The final layer is a classification layer using a softmax function. This model has around 24.2 million trainable parameters.

2.3.3. Model#3: CNN-SLSTM-FC model

This model consists of CNN layers followed by stacked LSTM layers. The architecture of this model differs from model#2 in the number of neurons of the stacked LSTM layers and the usage of a fully connected (FC) layer that has been added after the stacked LSTM layers. The number of neurons in each LSTM layer is doubled, aka 1024 neurons and 512 neurons in the first and second LSTM layers, respectively. These layers are followed by a FC layer with 256 neurons and the output of this layer is fed into a dropout layer followed by a classification layer. The proposed architecture has over 52 million trainable parameters.

3. Experiments and results

3.1. Sign language datasets

Three datasets have been used to evaluate the proposed models. These datasets contain signs from two different sign languages performed by several signers.

A) K-RSL dataset: Kazakh-Russian sign language (K-RSL) dataset [60] is a multisigner database that consists of 5200 video samples of 20 question signs (e.g., what, where (direction), where (location), who, which, how, how much, etc.) used in statements and questions [61]. Each sign is repeated around 260 times by five signers (40 samples for four signers and 100 samples for one signer). The duration of each sign differs from sign to sign and sometimes from signer to signev to have the number of frames between 3 and 60. A samples from this dataset is shown in Figure 3a.

B) Shanableh datset: We also used an ArSL database proposed by Shanableh and Assaleh [62] to evaluate the proposed models. This dataset consists of 23 dynamic signs performed by three signers. Each signer repeated each sign 50 times to end up with 3450 samples. The duration of each sample differs based on the sign and signer and the range of frames of this database is between 2 and 13 frames. The dataset was recorded in unconstrained environment using a video camera and samples of this dataset are shown in Figure 3b.

C) KArSL dataset: This is another ArSL database that consists of 502 isolated signs (letters, numbers and words) [63]. Each sign was performed by three signers and repeated 50 times. This dataset was collected using multimodal Kinect V2 sensor that provides three types of information for each captured sign. It provides color video, depth, and joint points information. This dataset is recorded at a rate of 30 frames per second to have around 10 to 150 frames per sign depending on the sign duration. We started experiments with 33 dynamic gestures selected randomly. We refer to this dataset in this manuscript as *KArSL-33*. Then, we analyzed the scalability of the proposed models with more signs of this database. We evaluated the models on 100 and 190 signs of this dataset and we refer to these datasets as *KArSL-100* and *KArSL-190*, respectively. We selected this number of signs since they were used by other published works. These signs are composed of static and dynamic gestures of ArSL of ArSL and a sample is shown in Figure 3c.

3.2. Experimental results

Several experiments have been conducted to evaluate the accuracy, scalability, and computation time of the proposed models on the three datasets discussed before. We run all the experiments using Tensorflow 2.4 on a workstation with Nvidia GeForce RTX 2080 TI GPU with 11 GB GPU memory and 64 RAM memory.

3.2.1. Accuracy and convergence analysis

In the first set of experiments, we analyzed various performance metrics of the proposed three models compared to the baseline model on the K-RSL dataset. We used 5-fold crossvalidation (CV) for each model and computed the average accuracy, precision, recall, and F1 score; the results are shown in Table 1. We also conducted statistical analysis by repeating each experiments 5 times and computed the mean and standard deviation and results are shown in Table 2. As shown in these tables, the proposed models have higher performance in all metrics compared to the baseline model. Moreover, the performance with raw input is better than using AFD for this dataset. Also the models have low variance as indicated by the standard deviation in Table 2.

Figure 3. Sample signs from the used sign language databases. (a) K-RSL, (b) Shanableh, (c) KArSL.

In another set of experiments, we used two datasets, namely K-RSL and Shanableh, to evaluate the accuracy of the proposed models under two different settings. The first setting is to test the model on the gestures performed by the same signer involved in the model training. The second setting is to test the model on gestures performed by all signers without any clue about the signer identity and we refer to this setting by *All* in the result tables. Table 3 shows the obtained accuracies using the proposed models. As shown in the table, using CNN to extract the spatial features and feed them into LSTM (model#1) improved the accuracy with all the evaluated datasets compared with the base model when the raw colored frame images were fed into these models. However, stacking LSTM layers with CNN (models#2 and #3) did not improve the accuracy significantly with raw colored frames. It is also noticeable that the models' accuracies with signer 5 of K-RSL dataset are less than other signers and this can be attributed to variance between the signs' samples performed by signer 5. For example, the samples of the sign "WHERE" have inconsistent motion among the recorded videos. This usually happens with an unprofessional signer who was unable to repeat the sign without significant variance. The corresponding changes in the loss functions with all signers using model#2 using early stopping technique with patient of 20 epochs are shown in Figures 4a and 4b, which demonstrate fast converge for K-RSL and Shanableh datasets, respectively.

We also evaluated the proposed models using AFD frame images. These frames are computed from the absolute difference between consecutive raw colored frame images. More information about this input representation can be found in subsection 2.1. As shown in Table 3, using AFD improved the accuracy of the base model significantly compared with other models. The base model with raw colored frames learns some

Table 1. 5-fold crossvalidation performance results in terms of average precision, recall, F1-score and accuracy of various models for the K-RSL dataset.

	Model	Precision	Recall	F1-score	Accuracy
AFD	**Baseline**	0.989	0.989	0.989	0.989
	Model#1	0.986	0.986	0.986	0.986
	Model#2	0.984	0.983	0.983	0.983
	Model#3	0.986	0.985	0.985	0.985
Raw	**Baseline**	0.918	0.909	0.908	0.910
	Model#1	0.988	0.987	0.987	0.987
	Model#2	0.991	0.991	0.991	0.991
	Model#3	0.990	0.990	0.990	0.990

Table 2. Statistical analysis by repeating each experiment 5 times on the K-RSL dataset and computing mean and standard deviation.

	Model	Precision	Recall	F1-score	Accuracy
AFD	**Baseline**	0.9307 ± 0.0259	0.9278 ± 0.028	0.9278 ± 0.0275	0.9288 ± 0.0275
	Model#1	0.9482 ± 0.0165	0.9456 ± 0.0176	0.9458 ± 0.0202	0.9465 ± 0.0198
	Model#2	0.9389 ± 0.0151	0.9376 ± 0.0159	0.9373 ± 0.0183	0.9382 ± 0.018
	Model#3	0.9473 ± 0.0070	0.9462 ± 0.0070	0.9459 ± 0.0105	0.9465 ± 0.0111
Raw	**Baseline**	0.8658 ± 0.0255	0.8476 ± 0.028	0.8472 ± 0.0283	0.8506 ± 0.0272
	Model#1	0.9598 ± 0.0157	0.9576 ± 0.0168	0.9578 ± 0.0194	0.9584 ± 0.0192
	Model#2	0.9661 ± 0.0081	0.9642 ± 0.0084	0.9645 ± 0.0119	0.9649 ± 0.0118
	Model#3	0.9678 ± 0.0158	0.9662 ± 0.0157	0.9664 ± 0.0188	0.9668 ± 0.0188

Table 3. Recognition accuracies.

Dataset	**Signer**	**Raw frames**				**AFD frames**			
		Base model	**Model#1**	**Model#2**	**Model#3**	**Base model**	**Model#1**	**Model#2**	**Model#3**
K-RSL	**1**	0.906	0.964	0.964	0.921	0.935	0.906	0.914	0.935
	2	0.889	0.968	0.968	0.926	0.968	0.921	0.937	0.952
	3	0.974	0.989	0.989	0.947	0.937	0.921	0.942	0.958
	4	0.738	0.940	0.957	0.888	0.963	0.897	0.897	0.916
	5	0.890	0.547	0.564	0.780	0.963	0.761	0.846	0.817
	All	0.815	0.953	0.957	0.940	0.945	0.929	0.924	0.936
Shanableh	**1**	1.0	0.997	0.997	1.0	0.994	0.983	0.997	0.994
	2	0.988	0.988	0.991	0.991	0.986	0.983	0.983	0.997
	3	1.0	1.0	1.0	1.0	0.994	0.991	0.988	0.988
	All	0.999	0.999	1.0	1.0	0.991	0.994	0.996	0.994

features not related to the gesture such as signer face and body. These static features are discarded using AFD technique which can justify the improvement in the accuracy of the base model using this type of input representation. In contrast, feeding the AFD data to other models did not improve the accuracy compared with raw colored frames. These models depend on extracting spatial features from the input frames that may be lost when applying AFD.

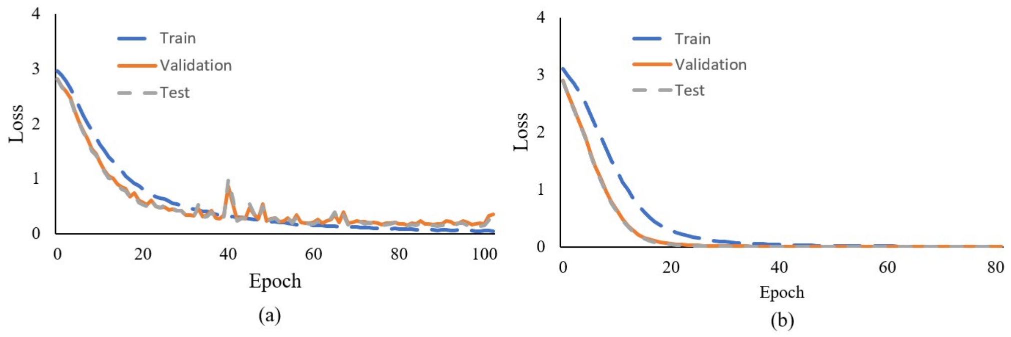

Figure 4. Loss functions of model#2 using early-stopping technique (Note: the test curves overlap with the validation curves due to minor loss differences): (a) K-RSL, and (b) Shanableh.

3.2.2. Scalability analysis

To evaluate the scalability of the proposed models, we studied the performance of the proposed models on different number of signs from the KArSL dataset. We initially randomly selected 33 signs of this dataset to evaluate the proposed models. Then, we did a scalability analysis with more signs of this dataset to show the performance of our models as we increase the number of signs from the KArSL dataset. We extended the number of signs to 100 and 190 signs. We selected these numbers of signs to be able to compare with other published works. Table 4 shows the accuracy of the proposed models with KArSL-33, KArSL-100 and KArSL-190 datasets using raw and AFD frames. The corresponding loss functions using Model#2 with all signers are shown in Figures 5a–Figure 5c, respectively. As shown in the table, the accuracy of the models does not decrease significantly as the number of signs increases. It is clear from the table that the misclassifications resulted mainly from numbers and letters signs. These signs are similar and the differences between some signs are on the finger shape or hand orientation which make them too confusing for recognition systems. Figures 6a and 6b show two of the signs that are misclassified. They differ only in using one finger as a part of the motion while the whole gesture is identical.

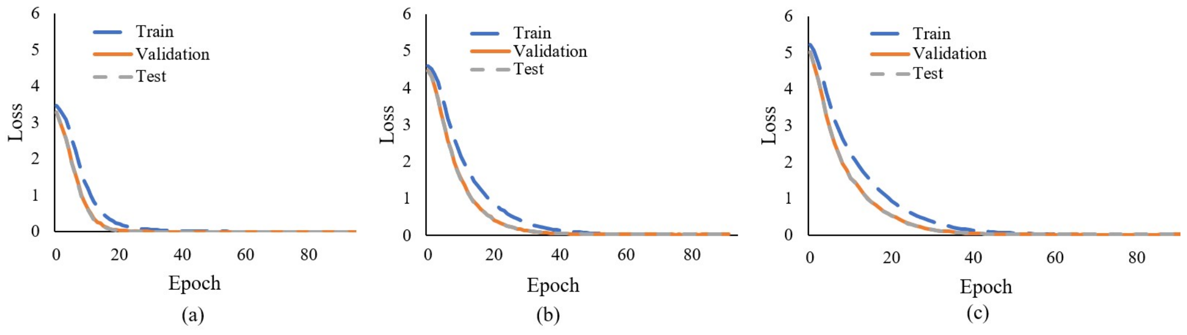

Figure 5. Loss functions of model#2 using early-stopping technique (Note: the test curves overlap with the validation curves due to minor loss differences): (a) KArSL-33, (b) KArSL-100, and (c) KArSL-190.

3.2.3. Computation time comparisons

In this part, we are evaluating the computation time of the proposed models on the three datasets. We started by evaluating the impact of the frame resolution on the accuracy and time of the model. The resolution of sign frames of the used datasets are different. KArSL dataset has the highest frame resolution compared with

Table 4. Scalability analysis of recognition accuracies for the proposed models on KArSL-33, KArSL-100 and KArSL-190 for various classes of the signs (KArSL-33 contains only words).

Dataset	**Signer**	**Class**	**Raw frames**				**AFD frames**			
			Baseline	**Model#1**	**Model#2**	**Model#3**	**Baseline**	**Model#1**	**Model#2**	**Model#3**
KArSL-33	**1**	**Words**	**1.0**	**0.996**	**1.0**	**0.996**	**1.0**	**1.0**	**1.0**	**1.0**
	2	**Words**	**1.0**	**1.0**	**1.0**	**1.0**	**0.996**	**0.992**	**0.992**	**0.992**
	3	**Words**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	All	**Words**	**1.0**	**1.0**	**1.0**	**1.0**	**0.994**	**0.997**	**0.997**	**0.996**
KArSL-100	**1**	**Numbers**	0.95	0.95	0.95	0.95	0.91	0.94	0.94	0.94
		Letters	0.99	1.0	1.00	0.99	0.99	0.97	0.96	0.96
		Words	1.00	1.0	1.00	1.00	0.98	1.00	1.00	1.00
		Average	**0.980**	**0.983**	**0.983**	**0.980**	**0.960**	**0.970**	**0.967**	**0.967**
	2	**Numbers**	0.99	1.0	0.99	0.99	0.95	0.96	0.95	0.96
		Letters	0.98	1.0	0.99	0.99	0.92	0.97	0.93	0.94
		Words	1.00	1.0	1.00	1.00	1.00	1.00	1.00	1.00
		Average	**0.990**	**1.0**	**0.993**	**0.993**	**0.957**	**0.977**	**0.960**	**0.967**
	3	**Numbers**	0.98	1.0	1.00	1.00	0.99	0.96	0.97	0.98
		Letters	1.00	1.0	1.00	1.00	0.99	0.99	0.98	0.99
		Words	1.00	1.0	1.00	1.00	1.00	1.00	1.00	1.00
		Average	**0.993**	**1.0**	**1.0**	**1.0**	**0.993**	**0.983**	**0.983**	**0.990**
	All	**Numbers**	0.96	0.98	0.97	0.97	0.95	0.95	0.96	0.95
		Letters	0.98	0.99	0.99	1.00	0.93	0.97	0.97	0.96
		Words	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		Average	**0.980**	**0.990**	**0.987**	**0.990**	**0.960**	**0.973**	**0.977**	**0.970**
KArSL-190	**1**	**Numbers**	0.94	0.94	0.95	0.95	0.94	0.94	0.95	0.94
		Letters	0.97	0.97	0.98	0.97	0.95	0.96	0.94	0.98
		Words	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		Average	**0.970**	**0.970**	**0.977**	**0.974**	**0.964**	**0.967**	**0.964**	**0.974**
	2	**Numbers**	0.96	1.00	1.00	1.00	0.93	0.95	0.94	0.96
		Letters	0.99	1.00	1.00	1.00	0.95	0.97	0.95	0.95
		Words	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00
		Average	**0.984**	**1.0**	**1.0**	**1.0**	**0.960**	**0.970**	**0.963**	**0.970**
	3	**Numbers**	0.97	1.00	0.99	0.99	0.98	0.98	0.97	0.96
		Letters	0.97	0.99	0.99	0.99	0.97	0.99	0.99	0.98
		Words	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		Average	**0.980**	**0.994**	**0.994**	**0.994**	**0.984**	**0.990**	**0.987**	**0.980**
	All	**Numbers**	0.96	0.98	0.98	0.98	0.95	0955	0.96	0.96
		Letters	0.99	0.99	0.99	0.99	0.96	0.971	0.98	0.97
		Words	1.0	1.0	0.99	1.0	1.0	0.997	1.0	1.0
		Average	**0.984**	**0.990**	**0.987**	**0.990**	**0.970**	**0.974**	**0.980**	**0.976**

the other two datasets used in this work. Therefore, we selected this dataset to evaluate the impact of frame resolution on the model performance. We started by experimenting the raw colored frames as captured by

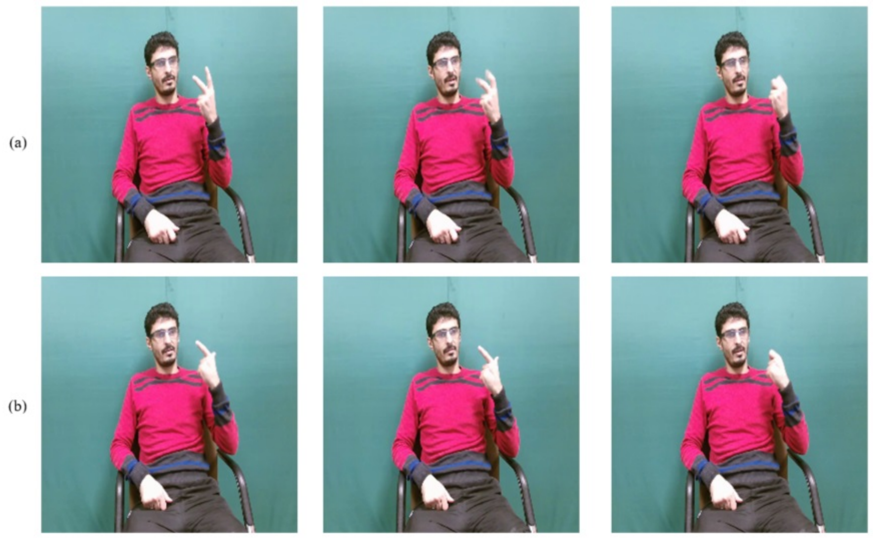

Figure 6. Three frames of two similar signs that are classified: (a) 700 sign, (b) 800 sign.

Kinect sensor where each frame has the resolution of 1920×1080 pixels. We cropped the colored frames to keep only the signer body. We used the joint points of the signer's head and hands retrieved by the Kinect sensor as reference points to segment the signer body and crop it into a resolution of 256×256. This cropping reduced the training time and improved the model accuracy.

We also compared the computation time of each model with all datasets on the two input representations, raw and AFD. Figures 7a–7d show the average training times of each model. As shown in these plots, the highest training time with all datasets except Shanableh dataset was with the baseline model when the raw colored frames were used as input to the models while model#2 has the lowest training time with all datasets except KArSL-33 dataset (Figures 7a and 7c). This can be attributed to the spatial features that are extracted using CNN that help in reducing the learning time of the LSTM. Figures 7b and 7d show the training time when AFD frames were used as input to the proposed models. This confirms that the baseline model has the lowest training time while model#1 has the highest training time with all datasets except KArSL-33 dataset (Figurs 7b and 7d). This can be attributed to the AFD input representation that causes losing the spatial features and consequently CNN may need more time to extract distinguishable features.

3.3. Comparison with other works

We compared the proposed models on publicly available datasets and compared the results with published work on the same datasets as described in subsection 3.1. Table 5 shows the accuracy reported by other works along with our work (the last eight lines). We did the comparison using the raw colored frames and AFD data representations. As shown in the table, our models outperformed other methods with all datasets. This performance did not degrade with the change in data representation which shows the efficiency of the proposed models. It is also noticeable that stacked LSTM (model#2) has the highest accuracy compared with other models. Since the three models are variations of the same proposed methodology, they have consistent performance to handle different datasets even when increasing the number of signs from 33 to 190. There were minor differences in their results which can be attributed to fact that these models capture similar low-level features due to utilizing similar architectures in the early layers.

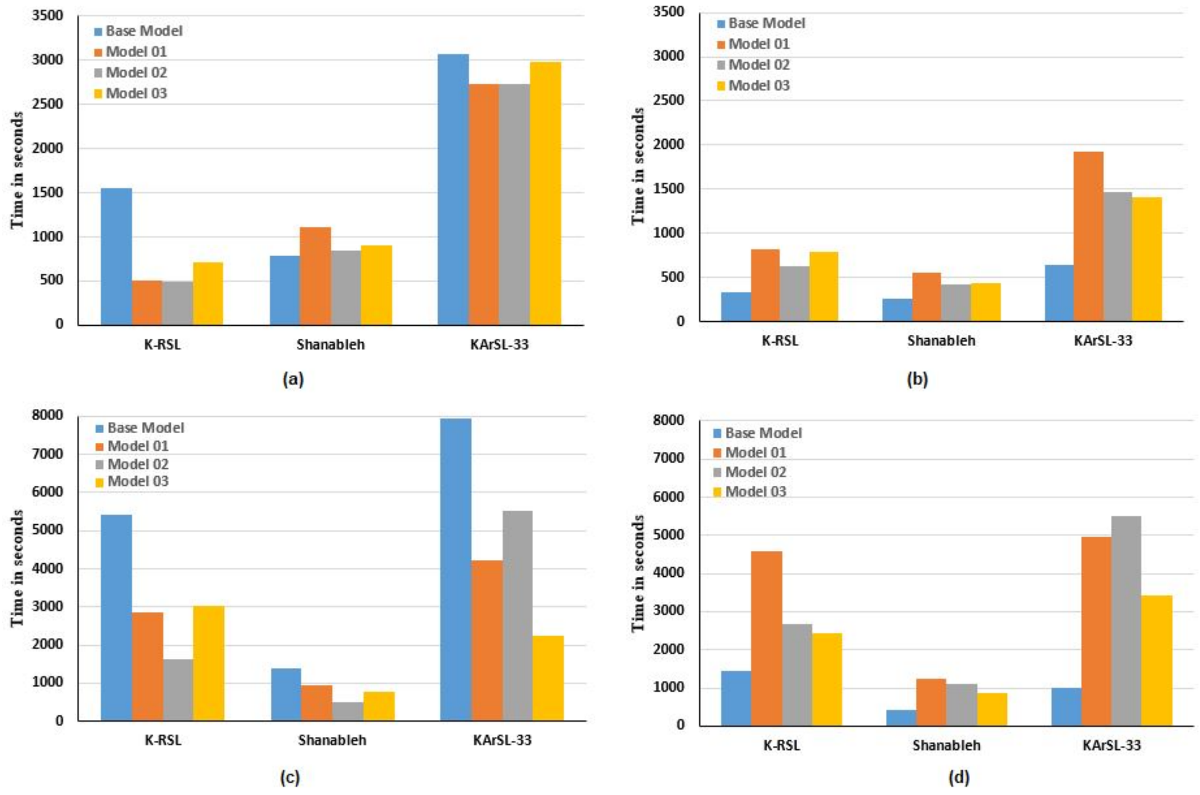

Figure 7. Training time of the proposed models: (a) Average of individual signers with raw colored frames, (b) average of individual signers with AFD frames, (c) all signers with raw colored frames, (d) all signers with AFD frames.

Table 5. Comparison with published work.

Method		**K-RSL**	**Shanableh**	**KArSL-100**	**KArSL-190**
Mukushev et al. [61]		0.782	-	-	-
DCT [62]		-	0.95	-	-
Hartley Transform [1]		-	0.99	-	-
Trajectory-based [64]		-		0.997	-
HOG+HMM [63]		-	-	-	0.881
AFD	Baseline	0.945	0.991	0.961	0.98
	Model#1	0.929	0.994	0.972	0.986
	Model#2	0.924	0.996	0.975	0.986
	Model#3	0.936	0.994	0.966	0.985
Raw	Baseline	0.815	0.999	0.985	0.990
	Model#1	0.953	0.999	0.989	0.994
	Model#2	**0.957**	**1.0**	**0.990**	**0.994**
	Model#3	0.940	1.0	0.989	0.994

4. Conclusion

With the advances in computer vision and deep learning, several applications have emerged in recent years. This paper proposed three models based on combining deep convolutional neural networks with long short-term memory for recognizing dynamic sign language gestures. The proposed cascaded models take advantage of the strengths of CNN and LSTM in leaning spatial and temporal feature, respectively, pertaining to the class of each sign language gesture. Several experiments have been carried out to evaluate and compare the performance and scalability of the proposed models using two variants of inputs of three datasets of two sign languages. The evaluation also considered the generalization of signer-specific models to crosssigner recognition. As a future work, other models and datasets can be used for comparison. Moreover, more work is required to improve the training time and extend the proposed approach to continuous sign language recognition.

Acknowledgment

The authors would like to acknowledge the support provided by King Fahd University of Petroleum and Minerals (KFUPM) for funding this work through project number SR191014.

References

[1] Sidig A, Luqman H, Mahmoud S. Transform-based arabic sign language recognition. Procedia Computer Science 2017; 117: 2–9.

[2] BinMakhashen G, Luqman H, El-Alfy E. Using gabor filter bank with downsampling and SVM for visual sign language alphabet recognition. In: 2nd Smart Cities Symposium (SCS 2019), IET Conference Proceedings; 2019. pp. 1–6.

[3] Nair A, Bindu V. A review on indian sign language recognition. International Journal of Computer Applications 2013; 73: 33–38.

[4] Cheok M, Omar Z, Jaward M. A review of hand gesture and sign language recognition techniques. International Journal of Machine Learning and Cybernetics 2019; 10: 131–153.

[5] Pisharady P, Saerbeck M. Recent methods and databases in vision-based hand gesture recognition: A review. Computer Vision and Image Understanding 2015; 141: 152–165.

[6] Luqman H, Mahmoud S. Automatic translation of arabic text-to-arabic sign language. Universal Access in the Information Society 2019; 18: 939–951.

[7] Bheda V, Radpour D. Using deep convolutional networks for gesture recognition in american sign language. arXiv preprint arXiv:1710.06836 2017; 1–5.

[8] Ranga V, Yadav N, Garg P. American sign language fingerspelling using hybrid discrete wavelet transform-gabor filter and convolutional neural network. Journal of Engineering Science and Technology 2018; 13: 2655–2669.

[9] Munib Q, Habeeb M, Takruri B, Al-Malik H. American sign language (ASL) recognition based on Hough transform and neural networks. Expert systems with Applications 2007; 32: 24–37.

[10] Zamani M, Kanan H. Saliency based alphabet and numbers of American sign language recognition using linear feature extraction. In: Proceedings of the 4th International Conference on Computer and Knowledge Engineering (ICCKE); 2014. pp. 398–403.

[11] Pan T, Lo L, Yeh C, Li J, Liu H et al. Real-time sign language recognition in complex background scene based on a hierarchical clustering classification method. In: Proceedings of IEEE Second International Conference on Multimedia Big Data (BigMM); 2016. pp. 64–67.

[12] Aowal M, Zaman A, Rahman S, Hatzinakos D. Static hand gesture recognition using discriminative 2d zernike moments. In: Proceedings of TENCON IEEE Region 10 Conference; 2014. pp. 1–5.

[13] Sabhara R, Lee C, Lim K. Comparative study of hu moments and zernike moments in object recognition. The Smart Computing Review 2013; 3: 166–173.

[14] Otiniano-Rodrıguez K, Cámara-Chávez G, Menotti D. Hu and zernike moments for sign language recognition. In: Proceedings of International Conference on Image Processing, Computer vision, and Pattern Recognition; 2012. pp. 1–5.

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

[16] Almeida S, Guimarães F, Ramírez, J. Feature extraction in Brazilian Sign Language Recognition based on phonological structure and using RGB-D sensors. Expert Systems with Applications 2014; 41: 7259–7271.

[17] Tolba M, Samir A, Abul-Ela M. 3D Arabic sign language recognition using linear combination of multiple 2d views. In: Proceedings of 8th International Conference on Informatics and Systems (INFOS); 2012. pp. 1–6.

[18] Ng C, Ranganath S. Real-time gesture recognition system and application. Image and Vision Computing 2002; 20: 993–1007.

[19] Pattanaworapan K, Chamnongthai K, Guo J. Signer-independence finger alphabet recognition using discrete wavelet transform and area level run lengths. Journal of Visual Communication and Image Representation 2016; 38: 658–677.

[20] Thalange A, Dixit S. COHST and wavelet features based Static ASL numbers recognition. Procedia Computer Science 2016; 92: 455–460.

[21] Shanableh T, Assaleh K, Al-Rousan M. Spatio-temporal feature-extraction techniques for isolated gesture recognition in Arabic sign language. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 2007; 37: 641–650.

[22] Zaki M, Shaheen S. Sign language recognition using a combination of new vision based features. Pattern Recognition Letters 2011; 32: 572–577.

[23] Hartanto R, Kartikasari A. Android based real-time static Indonesian sign language recognition system prototype. In: Proceedings of IEEE 8th International Conference on Information Technology and Electrical Engineering (ICITEE); 2016. pp. 1–6.

[24] Celebi S, Aydin A, Temiz T, Arici T. Gesture recognition using skeleton data with weighted dynamic time warping. In: Proceedings of International Conference on Computer Vision Theory and Applications (VISAPP); 2013. pp. 620–625.

[25] Plouffe G, Cretu A. Static and dynamic hand gesture recognition in depth data using dynamic time warping. IEEE Transactions on Instrumentation and Measurement 2015; 65: 305–316.

[26] Jin C, Omar Z, Jaward M. A mobile application of American sign language translation via image processing algorithms. In: Proceedings of IEEE Region 10 Symposium (TENSYMP); 2016. pp. 104–109.

[27] Kumar B, Manjunatha M. A hybrid gesture recognition method for American sign language. Indian Journal of Science and Technology 2017; 10: 1–12.

[28] Cui R, Liu H, Zhang C. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition; 2017. pp. 7361–7369.

[29] Huang J, Zhou W, Li H, Li W. Sign language recognition using 3d convolutional neural networks. In: Proceedings of IEEE International Conference on Multimedia and Expo (ICME); 2015. pp. 1–6.

[30] Li S, Yu B, Wu W, Su S, Ji R. Feature learning based on sae–pca network for human gesture recognition in rgbd images. Neurocomputing 2015; 151: 565–573.

[31] Barros P, Magg S, Weber C, Wermter S. A multichannel convolutional neural network for hand posture recognition. In: Proceedings of International Conference on Artificial Neural Networks, Springer; 2014. pp. 403–410.

[32] Prasuhn L, Oyamada Y, Mochizuki Y, Ishikawa H. A HOG-based hand gesture recognition system on a mobile device. In: Proceedings of IEEE International Conference on Image Processing (ICIP); 2014. pp. 3973–3977.

[33] Azar S, Seyedarabi H. Trajectory-based recognition of dynamic Persian sign language using hidden Markov model. Computer Speech & Language 2020; 61: 101053.

[34] Sidig A, Mahmoud S. Trajectory based arabic sign language recognition. International Journal of Advanced Computer Science and Applications 2018; 9 (4): 283–291.

[35] Liu T, Zhou W, Li H. Sign language recognition with long short-term memory. In: Proceedings of IEEE International Conference on Image Processing (ICIP); 2016; pp. 2871–2875.

[36] Sabyrov A, Mukushev M, Kimmelman V. Towards real-time sign language interpreting robot: evaluation of non-manual components on recognition accuracy. In: Proceedings of CVPR Workshops; 2019.

[37] Elons A, Ahmed M, Shedid H. Facial expressions recognition for arabic sign language translation. In: Proceedings of 9th IEEE International Conference on Computer Engineering & Systems (ICCES); 2014. pp. 330–335.

[38] Luqman H, El-Alfy E. Towards hybrid multimodal manual and nNon-Manual arabic sign language recognition: mArSL database and pilot study. Electronics 2021; 10: 1739.

[39] Rao G, Kishore P. Selfie video based continuous Indian sign language recognition system. Ain Shams Engineering Journal 2018; 9: 1929–1939.

[40] AL-Rousan M, Assaleh K, Tala'a A. Video-based signer-independent arabic sign language recognition using hidden markov models. Applied Soft Computing 2009; 9: 990–999.

[41] Kelly D, Reilly Delannoy J, Mc Donald J, Markham C. A framework for continuous multimodal sign language recognition. In: Proceedings of International Conference on Multimodal Interfaces; 2009. pp. 351–358.

[42] Von Agris U, Knorr M, Kraiss K. The significance of facial features for automatic sign language recognition. In: Proceedings of 8th IEEE International Conference on Automatic Face & Gesture Recognition; 2008. pp. 1–6.

[43] Sarkar S, Loeding B, Parashar A. Fusion of manual and non-manual information in american sign language recognition. In Handbook of Pattern Recognition and Computer Vision, World Scientific; 2010. pp. 477–495.

[44] Quesada L, Marín G, Guerrero L. Sign language recognition model combining non-manual markers and handshapes. In: Proceedings of International Conference on Ubiquitous Computing and Ambient Intelligence, Springer; 2016. pp. 400–405.

[45] Kumar P, Roy P, Dogra D. Independent bayesian classifier combination based sign language recognition using facial expression. Information Sciences 2018; 428: 30–48.

[46] Adaloglou N, Chatzis T, Papastratis I, Stergioulas A, Papadopoulos G et al. A comprehensive study on sign language recognition methods. arXiv preprint arXiv:2007.12530 2021; v2: 1–14.

[47] Wang L, Xu Y, Cheng J, Xia H, Yin J et al. Human action recognition by learning spatio-temporal features with deep neural networks. IEEE Access 2018; 6: 17913–17922.

[48] Karaki H, Alomari S, Refai M. A comprehensive survey of the vehicle motion detection and tracking methods for aerial surveillance videos. International Journal of Computer Science and Network Security 2019; 19: 93–106.

[49] Gers F, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with lstm. Neural Computation 2000; 12: 2451–2471.

[50] Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J. A novel connectionist system for unconstrained handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 2008; 31: 855–868.

[51] Greff K, Srivastava R, Koutník J, Steunebrink B, Schmidhuber J. LSTM: a search space odyssey. IEEE Transactions on Neural Networks and Learning Systems 2016; 28: 2222–2232.

[52] Ozdemir M, Ozdemir G, Guren O. Classification of COVID-19 electrocardiograms by using hexaxial feature mapping and deep learning. BMC Medical Informatics and Decision Making 2021; 21: 1–20.

[53] Ozdemir M, Degirmenci M, Izci E, Akan, A. EEG-based emotion recognition with deep convolutional neural networks" Biomedical Engineering. Biomedical Engineering / Biomedizinische Technik 2021; 66 (1): 43–57.

[54] Chen Y, Zhu L, Ghamisi P, Jia X, Li G, Tang L. Hyperspectral images classification with Gabor filtering and convolutional neural network. IEEE Geoscience and Remote Sensing Letters 2017; 14: 2355–2359.

[55] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 2015.

[56] Santurkar S, Tsipras D, Ilyas A, Madry A. How does batch normalization help optimization? In: Advances in Neural Information Processing Systems; 2018. pp. 2483–2493.

[57] Hinton G, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 2012.

[58] Kingma D, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 2014.

[59] Goldberg Y. A primer on neural network models for natural language processing. Journal of Artificial Intelligence Research 2016; 57: 345–420.

[60] Imashev A, Mukushev M, Kimmelman V, Sandygulova A. A dataset for linguistic understanding, visual evaluation, and recognition of sign languages: the k-rsl. In: Proceedings of 24th Conference on Computational Natural Language Learning; 2020. pp. 631–640.

[61] Mukushev M, Sabyrov A, Imashev A, Koishybay K, Kimmelman V et al. Evaluation of manual and non-manual components for sign language recognition. In: Proceedings of 12th Language Resources and Evaluation Conference; 2020. pp. 6073–6078.

[62] Shanableh T, Assaleh K. User-independent recognition of Arabic sign language for facilitating communication with the deaf community. Digital Signal Processing 2011; 21: 535 – 542.

[63] Sidig A, Luqman H, Mahmoud S, Mohandes M. KArSL: arabic sign language database. Transactions on Asian and Low-Resource Language Information Processing 2021; 20(1): 1–19.

[64] Sidig A, Mahmoud S. Trajectory based arabic sign language recognition. International Journal of Advanced Computer Science and Applications 2018; 9: 283–291.