An Efficient Convolutional Neural Network Classification Model for Several Sign Language Alphabets

Ahmed Osman Mahmoud, Ibrahim Ziedan, Amr Ahmed Zamel
Computer and Systems Department, Faculty of Engineering, Zagazig Uni., Zagazig, Egypt

Abstract—**Although deaf people represent over 5% of the world's population, according to what the World Health Organization stated in May 2022, they suffer from social and economic marginalization. One way to improve the lives of deaf people is to try to make communication between them and others easier. Sign language, the means through which deaf people can communicate with other people, can benefit from modern techniques in machine learning. In this study, several convolutional neural networks (CNN) models are designed to develop an efficient model, in terms of accuracy and computational time, for the classification of different signs. This research presents a methodology for developing an efficient CNN architecture from scratch to classify multiple sign language alphabets, which has numerous advantages over other contemporary CNN models in terms of prediction time and accuracy. This framework analyses the effect of varying CNN hyper-parameters, such as kernel size, number of layers, and number of filters in each layer, and picks the ideal parameters for CNN model construction. In addition, the suggested CNN architecture operates directly on unprocessed data without the need for preprocessing to generalize it across other datasets. In addition, the capacity of the model to generalize to diverse sign languages is rigorously evaluated using three distinct sign language alphabets and five datasets, namely, Arabic (ArSL), two American English (ASL), Korean (KSL), and the combination of Arabic and American datasets. The proposed CNN architecture (SL-CNN) outperforms state-of-the-art CNN models and traditional machine learning models achieving an accuracy of 100%, 98.47%, 100%, and 99.5% for English, Arabic, Korean, and combined Arabic-English alphabets, respectively. The prediction or inference time of the model is about three milliseconds on average, making it suitable for real-time applications. So, in the future, it is easy to turn this model into a mobile application.**

Keywords*—*Convolutional neural network (CNN); sign language; Arabic sign language (ArSL); American sign language (ASL); Korean sign language (KSL); Complexity time

I. Introduction

The World Health Organization (WHO) stated in May 2022 that there are more than 360 million deaf people around the world. 80% of those who are deaf live in developing countries and use more than 300 sign languages [1]. Deaf people who suffer from hearing problems have trouble in their daily lives communicating with each other and with other people. Also, they have lower chances of having an adequate level of education.

Sign language is the means of communication between deaf people and other people and consists of hand signs and gestures for spelling letters and words. In the last two decades, many researchers have investigated several machine learning models for developing several sign language recognition models, e.g., Arabic [2], American [3], Korean [4], Indian, Chinese, and others [5]. The approaches for sign language recognition can be classified into two main categories: sensor-based and vision-based [6]. In a sensor-based, the speaker wears gloves or sensors, and the movement and body orientation are translated into a time series of sensor readings depending on the word or letter sign. Within the same category, several researchers use Microsoft Kinect [7], developed by Microsoft, for sign classification without wearing gloves or sensors. Microsoft Kinect has three optical sensors. It provides three outputs: an RGB image, an infrared (IR) image, or a depth image, and defines up to 25 skeleton joints. Generally, the other hand, in the vision-based approach [8], images are taken by a camera and analyzed to determine the shape of signs intended by the speaker. In this approach, a researchers use depth images and skeleton joints to analyze the kinematic movement of the body or hand to determine the word or alphabet character. In this way, sign classification became easier. On sufficient number of image examples for each sign should be collected to improve classification performance.

Several methods for sign language recognition, utilizing both traditional and deep learning techniques, have been proposed in the literature [5]. Traditional techniques such as Support Vector Machine (SVM) [9], Hidden Markov Model (HMM) [10], and Random Forest (RF) [11] have all been tried by many researchers for classifying sign language alphabet recognition, but they have all yielded unsatisfactory results. On the other hand, recent studies have shown that the CNN model is one of the most commonly used models in sign language recognition [6]. Surveys such as those conducted by Rastgoo et al. have shown that many models have been suggested by various researchers for sign language recognition with the help of deep learning techniques [5].

A. Motivation

Many researchers have tried traditional machine learning methods for classifying sign language alphabet recognition models [12], but they have not provided satisfactory results. Recently, the CNN model has been widely utilized in sign language recognition, but it has not offered an efficient CNN architecture, which is considerably more challenging due to CNN's numerous hyper-parameters. Moreover, the prediction time, which is the time to predict a single sign and a critical factor in practice, is usually ignored. In addition, although CNN model hyper-parameters have a large effect on performance and accuracy [13], they are not fully investigated. All these issues may affect model generalization for newly unseen data. These are the primary motivations for us to study the effect of selecting hyper-parameters in CNN and how we can generalize the CNN architecture across several datasets. The primary objective of this study is to provide a realistic, straightforward, and efficient CNN architecture. In addition, it can work on several sign language alphabets.

B. Contribution

In this paper, several CNN models are designed and analyzed to develop an optimal model for sign language recognition without any preprocessing.

- This research provided a method for selecting an optimal CNN model by analyzing the results of various hyper-parameters such as kernel size, number of layers, and number of filters in each layer.
- In the beginning, this approach investigated how changing the layer and filter numbers impacted accuracy.
- Then, the model with the highest accuracy and the fastest prediction time is chosen.
- In the end, we incorporate the impact of kernel size and the number of hidden layers to pick the best candidate.
- This approach operates directly on unprocessed data to generalize it across several datasets.
- The proposed model is examined on four different datasets, Arabic, American 2018, American 2012, and Korean alphabets, to investigate its robustness against data variation.
- In addition, the model is also tested on a combined dataset of Arabic and American alphabets.

This paper is organized as follows: Section II provides a literature review on Arabic, English, and Korean sign language detection. In Section III, a detailed design of the proposed CNN model is presented, including the selection of optimal hyper-parameters for Arabic and English sign languages. The proposed model is presented in Section IV. Several experiments are then conducted in Section V to evaluate and compare the proposed model to other state-of-the-art classification models for three sign language alphabets. Finally, conclusions and ideas for extending the current work are drawn in Section VI.

II. Literature Review

Sign language differs from country to country. For example, there are Arabic, American, and Korean alphabets. In the past decade, several research efforts have been made to automate sign language processing. Some researchers used traditional classifiers, while others used convolutional neural networks or recurrent neural networks (RNNs).

Concerning Arabic sign language (ArSL) recognition, Alzohairi, Alghonaim, et al. [9], for example, presented an SVM model to classify ArSL images. The authors, using a smartphone camera, collected a dataset of 900 images for 30 alphabet characters and extracted features from images using the histogram of oriented gradient (HOG) descriptor. The model achieved a low accuracy of 63.5%. In addition, Hasasneh and Taqatqa [14] proposed a model based on a restricted Boltzmann machine and tiny images for 39 Arabic alphabetic sign language groups.

Convolutional neural networks have also been applied for ArSL recognition. For example, Alani and Cosma [15] proposed two CNN models consisting of seven convolutional layers and four pooling layers for the ArSL 2018 dataset. The first one achieved an accuracy of 96.59%, while the second one used some sampling techniques to improve the accuracy to 97.29%. Also, Elsayed and Fathy [16] also implemented a CNN containing five convolutional layers and three pooling models for the Arabic alphabet and word recognition. They used a premade dataset consisting of 54049 samples to evaluate their model and produced a low accuracy of 88.87% for alphabet pattern recognition.

On the other hand, several techniques for American Sign Language (ASL) classification have been extensively studied. This usually consists of a two-stage feature extraction and a classifier. For instance, Aly, Aly, et al. [17] developed an SVM model to classify the ASL alphabet using a dataset collected by Microsoft Kinect from several users. The collected data is preprocessed for segmentation and feature extraction by the principal component analysis network (PCANet). This model achieves an accuracy of 88.7%. Shin, Matsuoka, et al. [12] proposed a model using SVM and light gradient boosting machine (GBM) to classify the ASL alphabets using the Massey alphabets dataset and the Kaggle alphabets dataset. The results were 99.39% for Massey and 87.60% for Kaggle.

The CNN technique is also used for ASL classification. For example, Fierro and Perez [8] built two CNN models for sharing parameters and achieved 96% accuracy. This model was fed by samples taken from the Kaggle dataset, which contains 29 subclasses. In addition, Abdulhussein and Raheem [13] also built a CNN model for classifying 24 ASL characters after preprocessing input images using image resizing, converting images to grayscale, and edge detection. Their model achieved an accuracy of 99.3%, and the training time was shorter compared to its peers. Furthermore, Wardana, Rachmawati, et al. [18] proposed a CNN model for the Kaggle ASL dataset, achieving an accuracy of 99.81%. The data is divided 70% for training, and the remainder of the dataset is equally divided between validation and testing. Also, Can, Kaya, et al. [19] suggested a CNN model for colored natural ASL pictures and compared their accuracy with five well-known transfer learning models, including VGG16, VGG19, ResNet50, and DenseNet121, to obtain 99.91% superior to his peers.

Finally, for Korean alphabets (KSL), Na, Yang, et al. [20] presented an SVM model for classifying 31 signs consisting of 14, 10, and 7 consonants, vowels, and double vowels, respectively. This dataset was collected from 15 participants who wore gloves while taking images. The tri-axial accelerometer signals were used to segment the sign gesture while the user was performing it. This model achieves a segmentation accuracy of 98.9%, which is superior to its peers, which used multiple sensors for segmentation. This model achieved a mean recognition accuracy of 92.2% for the Korean alphabet. Multiclass SVM was designed with six different kernels (e.g., linear, quadratic, and cubic) and optimized through accuracy comparison. The quadratic kernel produced the best classification rate among the others. In contrast, Yeo and Shin [21] proposed a model for 12 classes of consonant and vowel letters. The model was designed by combining electromyography (EMG), accelerometers, and gyro sensors to build a multi-variate Gaussian model and maximum likelihood estimation through Bayesian theory for classification. As a result, accuracy rates of 99.13% and 99.97% were achieved for consonants and vowels, respectively.

Most researchers aimed to improve the performance of the CNN by studying the influence of preprocessing techniques such as filters [22] or other techniques (e.g., cropping, adding noise, and data normalization) [23]. Despite the preprocessing success in some cases, there is no way to generalize it for all datasets. Therefore, in the proposed model, preprocessing is not utilized to examine the power of the CNN, test the effect of its parameter in the original data, and test the model against dataset variation.

All the aforementioned traditional or modern techniques for sign language recognition models were trained and tested on a single dataset and ignored the diversity of various sign languages. In addition, some researchers used a small, collected dataset to train their models, so they could not be generalized. They also neglect prediction time, although it is a critical factor in real-time applications. Finally, even though the CNN technique was used, the CNN parameters were not studied sufficiently to achieve optimal accuracy and a reasonable prediction time. The main contribution of this paper can be summarized as follows: First, an efficient CNN sign language recognition is developed by studying the effect of CNN hyper-parameters to achieve higher accuracy compared with its peers using original data without any preprocessing. Second, the proposed model is trained on three sign language alphabets, namely, English, Arabic, and Korean, to evaluate its robustness against data variation. Third, the model is trained and tested to classify numerous sign languages by combining two different sign language datasets, namely Arabic and English. Finally, the prediction time is carefully considered during the design of the proposed model.

III. Methodology of CNN Hyper-parameters Selection

A CNN is one of the most popular techniques in deep learning (DL) that is successfully used in classification problems [24] and has high success rates in several problems such as hand gestures and object detection [25]. CNN consists of convolution, pooling, and fully connected layers. The convolution layers are the main layers in CNN, as they are responsible for creating a feature map using a set of filters whose number is a design parameter, or hyper-parameter [26]. Convolution is a linear multiplication operation between the input image and a filter whose size (the kernel size is another hyper-parameter) is smaller than the input image [27]. This results in an activation map (feature map) whose size is less than the input image and whose count is the same as the number of filters used in this operation. The size of the feature map is calculated according to Eq. 1.

$$outputsize = \frac{N - N_f}{stride} + 1 \tag{1}$$

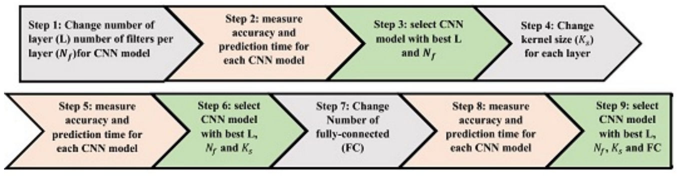
Step 1: Change number of layer (L) number of filters per layer (N_f)for CNN model
Step 2: measure accuracy and prediction time for each CNN model
Step 3: select CNN model with best L and N_f
Step 4: Change kernel size (K_s) for each layer
Step 5: measure accuracy and prediction time for each CNN model
Step 6: select CNN model with best L, N_f and K_s
Step 7: Change Number of fully-connected (FC)
Step 8: measure accuracy and prediction time for each CNN model
Step 9: select CNN model with best L, N_f, K_s and FC

Fig. 1. Flow diagram of the methodology framework for the selection of CNN hyper-parameters.

where N is the width or height of the input image, N_f is the filter's width or height, and the stride is the pixel size between each convolution [28]. The size of the feature map in Eq. 1 must be an integer number; otherwise, padding is employed. Padding is the process of increasing the image's size without changing its content. The input image, after being convolved repeatedly with filters, shrinks in volume spatially. Shrinking too fast is not good; it does not work well. Pooling combines the nearby units to reduce the input size for the next layer. It includes maximum pooling and average pooling. Using the pooling layer directly after the convolution layer is not necessary, but its type and location are also a hyper-parameter whose settings are set empirically using expertise. The result of repeated convolution and pooling is that the list of features (a vector of features) is the input for the last layer of CNN (the number of convolutional layers is a hyper-parameter). The fully connected layer classifies the input to the best class [29].

The multiple hyper-parameters that CNN uses to make the process of picking an architecture much more challenging [30]. In this section, a detailed study is conducted for the optimal selection of CNN model hyper-parameters, namely kernel size (K_s), the number of filters (N_f), the number of convolution layers (L), and the number of fully connected hidden layers, to achieve optimal accuracy in a reasonably short time for ArSL. The steps of the methodology are summarized in Fig. 1. Initially, this method looked at the effect of varying the layer and filter numbers on precision. Secondly, a model is selected based on its speed and accuracy of prediction. Finally, the influence of kernel size and the number of hidden layers is considered to select the optimal CNN architecture. The methodology is discussed in detail in the following subsections.

A. Selecting the Number of Layers and Filters Per Layer

The number of filters Nf in each convolution layer affects the test time because the more filters used, the more computational time is required. This occurs because the output of a convolution layer, i.e., the number of activations maps, equals the number of filters used in that layer [31]. The number of filters to be used in each layer is chosen according to Eq. 2.

$$N_f = 2^k; \qquad k = 2, 3, 4, 5, ... \tag{2}$$

to achieve the best accuracy within a reasonable prediction time. Also, the number of convolution layers has a vital role in the classification time in the ArSL problem. Because the image size was 64 by 64, the maximum number of convolutions with stride 2 would be 6. So, the number of layers is varied between 1 to 6, and tests are repeated 50 times to determine the best number of layers that gives the best average test time.

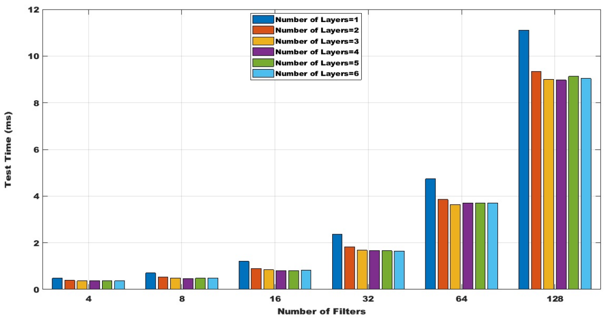

Fig. 2. Test time as a function of the number of layers and number of filters per layer.

TABLE I. Accuracy and Test Time for ArSL Using Different Numbers of Layers and Filters Per Layer

L	N_f=4		N_f=16		N_f=128	
N_f	Acc(%)	T_S (ms)	Acc (%)	T_S (ms)	Acc (%)	T_S (ms)
L = 2	95.4770	0.3793	96.8450	0.8819	96.586	8.9003
L = 4	95.2921	0.3606	97.6707	0.7986	98.52	8.8196
L = 6	56.6182	0.3598	96.1794	0.7969	98.5827	8.9013

Acc is the accuracy and T_S is the Test time in milliseconds //

In this study, several CNN models are built with different combinations of layers and filters per layer. The test time is estimated without training, as shown in Fig. 2. It can be realized that the test time increases with the number of filters per layer but decreases as the number of layers increases. This is because the number of extracted features from CNN (the last layer of CNN) decreases.

On the other hand, to take accuracy into account and decide the optimal number of layers and filters per layer, the CNN models with all combinations of layer numbers 2, 4, and 6 and filter numbers 4, 16, and 128 were trained. Table I shows the test time and accuracy in classifying the ArSL dataset. As can be seen from Table I, although a CNN with 4 filters per layer has the least amount of time, its accuracy is inferior compared to that obtained using 16 or 128 filters. Therefore, it is clear that the number of filters per layer should be at least 16. Considering the number of layers, it is noted that accuracy obtained using more than 4 layers either drops (N_f = 4 or 16) or is not significantly improved (N_f = 128). Based on these observations, the number of layers in the proposed model is set to 4. For this number of layers, the accuracy improved with the increase in the number of filters. Unfortunately, this comes at the expense of a significantly longer prediction time. Therefore, in the proposed model, instead of using 128 filters in all layers, the number of filters in the four layers is set respectively to 32, 64, 128, and 128. This reduces the total prediction time while maintaining acceptable accuracy at the last two layers.

B. Selecting the Kernel Size

Kernel size, which is usually taken as an odd number less than 10, also affects the features produced by the convolution [32]. Therefore, kernel sizes of 3, 5, 7, and 9 are trained for the ArSL classification problem. During this experiment, 10 runs are performed, and the average results are reported in Table II. As it can be seen, kernel size has a slight effect on the accuracy of training and testing. In contrast, the training time and test time increase nonlinearly with kernel size. Based on these observations, the optimal kernel size would be 3x3, as it achieves the minimum training and testing times while maintaining the same accuracy.

TABLE II. Training, Test Accuracy, Training, and Testing Time vs. CNN Kernel Size

K_s	T_r Acc (%)	T_s Acc (%)	T_r time (h)	T_S time (ms)	Std (T_s)
3x3	100	98.55	3.39	2.6621	4.4920 x 10^{-5}
5x5	100	98.71	5.02	3.5870	1.1804 x10^{-4}
7x7	100	98.87	6.62	8.0030	1.0546 x10^{-4}
9x9	100	98.77	9.27	11.8441	1.5197 x10^{-4}

T_r is training, T_s is test and std is the Standard deviation.

TABLE III. Test Accuracy, Training and Testing Time vs. CNN Fully Connected Hidden Layer

Hidden No.	T_r Acc(%)	T_s Acc(%)	T_r time(h)	T_s time(ms)	std(T_s)
0	100	98.3362	4.155	2.8157	3.0857 x 10^{-5}
1	100	98.5457	4.385	2.8760	3.3840 x 10^{-5}
2	100	98.7183	4.360	2.8893	3.4234 x 10^{-5}
3	100	98.5580	4.455	2.9107	1.7433 x 10^{-5}

T_r is training, T_s is test and std is the Standard deviation.

C. Selecting the Number of Fully Connected Hidden Layers

CNN always ends with a fully connected network that contains hidden layers whose number is considered a hyper-parameter [33]. The network with 0, 1, 2, and 3 hidden layers is trained in ArSL to investigate the best number of hidden layers. Each model is trained and tested ten times, and the average results are reported in Table III. As can be noted, fully connected two hidden layers achieve the best test accuracy with slightly higher test and training times. By adding the last output layer to the two hidden layers, three fully connected layers are considered in the proposed CNN model.

D. Study the Time Complexity of CNN's Prediction Time

In this section, the time complexity of CNN's prediction time is derived. The number of operations in a single convolution layer depends on the size of the image (N x N), the kernel size and the number of filters per layer. The number of convolutions is performed for each pixel in the image, so the time complexity is proportional to the total number of pixels in the image (N^2). Each pixel is multiplied by a window of size K_s x K_s. This is done for each filter; so, the time complexity for a single convolution layer is given by Eq. 3.

$$T_{conv} = O(N^2 K_s^2 N_f) \tag{3}$$

The feature map output from a single convolution layer is equal to the image multiplied by the number of filters in this layer. To reduce the dimension of the feature map, a Max-pooling layer is applied with a 2x2 window in the feature map of size N x N x N_f, whose time complexity can be expressed as Eq. 4.

$$T_{mp} = O(4N^2 N_f) \tag{4}$$

For L layers, the time complexity in Eq. 4 can therefore be written as Eq. 5.

$$T_L = O((N^2K_s^2N_f + 4N^2N_f)L) \tag{5}$$

where time complexity is linear in L. For fully connected layers, the feature maps are generated from the final convolution layer and the size of the input layer. The max-pooling layer decreases the size of the image by one-fourth (for a stride of 2). So, the final feature map size is $N^2/4^L$, and the time complexity of the first fully connected layer would be Eq. 6.

$$T_{FC} = O(\frac{N^2N_f}{4^L}) \tag{6}$$

So, the total time complexity would be calculated as shown in Eq. 7.

$$O((N^2K_s^2N_f + 4N^2N_f)L + \frac{N^2N_f}{4^L} + TimeofOtherFC) \tag{7}$$

The time for other fully connected layers is dropped in big-O notation from Eq. 7 as they are constant terms. Therefore, the time of fully contented dropped and all constants are dropped as shown in Eq. 8.

$$T = O(K_s^2N^2N_fL + \frac{N^2N_f}{4^L}) \tag{8}$$

From Eq. 8, it can be concluded that the time complexity of the CNN model is linearly proportional to Nf, the square of Ks, and nonlinear with L. Furthermore, to investigate the validity of Eq. 8, three examinations are made by building several CNN models, changing one parameter at a time, and averaging the test time over 50 runs. The procedure can be stated as follows:

- Varying the K_s from 1 to 19, while fixing the layer number at 4 and the number of filters at 16. Fig. 3 shows the measured test time as a function of the K_s. As it can be seen, the relationship can be fitted using a quadratic polynomial, which means that the time complexity of the CNN model is proportional to the square of K_s.
- Investigating the effect of the number of layers L on test time (at K_s = 3, N_f = 16). Fig. 4 shows that the time decreases as the number of layers increases up to 4 layers. Above 4 layers, the time starts to increase slightly. This experiment shows that the test time is inversely proportional to the number of layers, and a good fit can be obtained using a fifth-degree polynomial.
- Examining the effect of the number of filters Nf on test time (at K_s = 3, L=4). Fig. 5 shows that the test time is linearly proportional to the number of filters and can be fitted using a linear equation.

From Eq. 8 and the previous experiments of CNN, it is noted that the time complexity of the CNN model increases as the kernel size and number of filters increase. Moreover, the K_s significantly affects the test time because it is in proportion to the square of the K_s. On the other hand, the test time decreases as the number of layers increases.

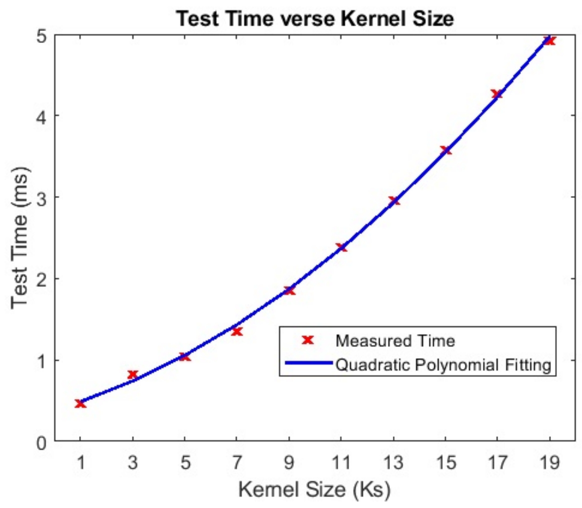

Fig. 3. Test time verse CNN K_s.

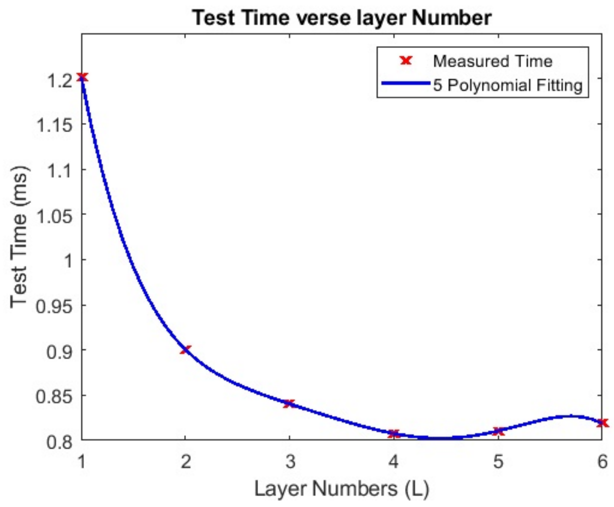

Fig. 4. Test time verse CNN layer numbers (L).

IV. The Proposed SL-CNN Model

In this section, the architecture of the proposed model is designed to achieve high performance and less prediction time. Also, to make the model operate directly on raw data to generalize it across several datasets and data robustness.The architecture of the proposed SL-CNN model is presented next and summarized as shown in Fig. 6. Based on the analysis of the experimental results in Section III, the proposed SL-CNN model is designed using four layers with a kernel size of 3 x 3. In addition, filter numbers ranging from 32 to 128 were also tested to achieve optimal accuracy and prediction times. The proposed SL-CNN model consists of four convolutional (Conv) layers, four max pooling (MP) layers, three fully connected layers, and two dropout layers, as shown in Fig. 7. The pooling layer has a Relu activation function. Additionally, 7 batch normalization (BN) layers are used to achieve a stable distribution of the activation values through training and normalizing the input layers [34]. Also, each convolution layer is followed by a Relu function layer.

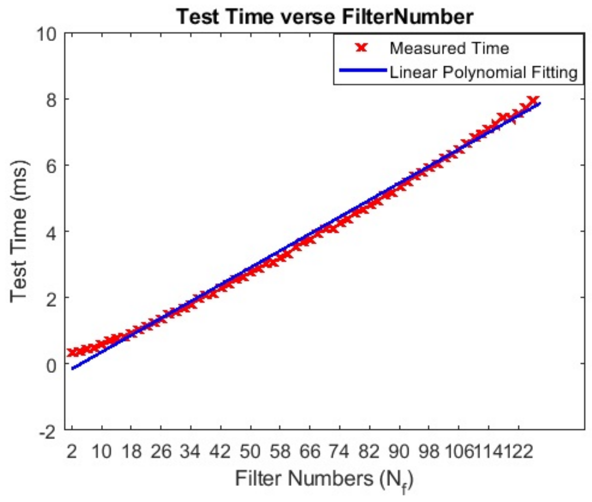

Fig. 5. Test time verse CNN filter numbers N_f.

After resizing the images, the first layer is a grey image of size 64 x 64. The second is the convolution layer that convolutes the input image with 32 filters whose sizes (3 x 3) produce 32 feature maps (FM) or activation maps, which equal the number of filters. Each activation map has the same dimension as the input image. The next layer is the batch normalization map, which accelerates deep network training by reducing internal covariate shifts [34]. Each convolution layer is followed by the Relu activation function. The fifth layer is max pooling with pool size (2 x 2), which decreases the activation map dimension to avoid the over-fitting problem and decreases the computation operation. The next two layers, layers 10 and 14, are the same as layer 2, but with filter sizes of 64, 128, and 128, respectively. Layers 7, 11, 15, 19, 23, and 27 are the same as layer 3. Layers 9, 13, and 17 have the same construction as layer 5.

The model ends with a fully connected neural network consisting of three layers. The first layer (layer 18) contains 1024 neurons and is activated by the Relu activation function. The dropout layer randomly sets input elements to zero with a 50% probability. The following layer is hidden in the fully connected network like the previous layer but with 512 neurons. The final layer is the output layer, which has X neurons that differ from one dataset to another (in the ArSL, ASL, KSL, and ArSL-ASL datasets, there are 32, 29, 14, and 61 neurons, respectively), representing the number of classes. The SoftMax activation function activates this layer. The SL-CNN model is learned in a supervised manner using the Adam optimizer for parameter optimization.

V. Experimental Results and Discussion

In this section, an extensive set of experiments is conducted to evaluate the performance of the proposed CNN model in sign language classification. First, three sign languages are considered, namely: Arabic sign language (ArSL), American English sign language (ASL), Korean sign language (KSL), and the combined Arabic-English sign languages.

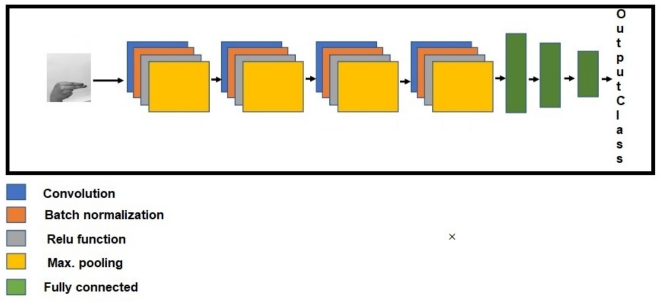

Fig. 6. The proposed structure of the SL-CNN model.

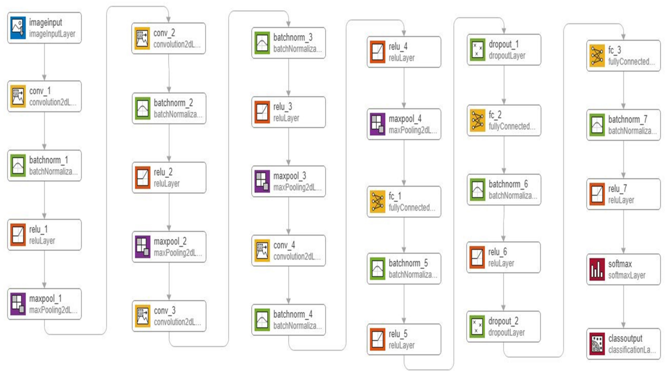
imageinput imageInputLayer
conv_1 convolution2dL...
batchnorm_1 batchNormaliza...
relu_1 reluLayer
maxpool_1 maxPooling2dL...
conv_2 convolution2dL...
batchnorm_2 batchNormaliza...
relu_2 reluLayer
maxpool_2 maxPooling2dL...
conv_3 convolution2dL...
batchnorm_3 batchNormaliza...
relu_3 reluLayer
maxpool_3 maxPooling2dL...
conv_4 convolution2dL...
batchnorm_4 batchNormaliza...
relu_4 reluLayer
maxpool_4 maxPooling2dL...
fc_1 fullyConnected...
batchnorm_5 batchNormaliza...
relu_5 reluLayer
dropout_1 dropoutLayer
fc_2 fullyConnected...
batchnorm_6 batchNormaliza...
relu_6 reluLayer
dropout_2 dropoutLayer
fc_3 fullyConnected...
batchnorm_7 batchNormaliza...
relu_7 reluLayer
softmax softmaxLayer
classoutput classificationLa...

Fig. 7. Layers details of the SL-CNN model.

A. Dataset Description

The datasets used in this work consist of a set of images, each of which corresponds to a single-sign character that is introduced to the classifier in a static manner. In this work, the sequential or dynamic case is not considered. In a classification problem, the datasets can be classified based on the distribution of images in each class in the training data into balanced and imbalanced datasets [35]. In balanced datasets, each class has the same number of training images. On the other hand, in imbalanced datasets, the number of training examples is not equal. The proposed SL-CNN model is trained and tested for three different sign language alphabets with the following five datasets: Arabic (imbalanced), two English (balanced), Korean (imbalanced), and combined Arabic-English (imbalanced). These sign language alphabets are briefly described next.

1) ArSL Alphabet: This dataset, also called ArSL 2018, contains 54,049 images of Arabic sign language alphabets with 32 characters. Fig. 8 shows a sample of ArSL 2018 and its classes. This dataset was gathered from 40 participants of various ages and is imbalanced, as illustrated in Fig. 9. Images have various dimensions and variances that may be removed using preprocessing techniques to eliminate noise and center the image [36]. The dataset is divided into 70% for training, and the remainder is divided equally between validation and testing, as shown in Table IV. The data is rearranged randomly for each experiment.

2) ASL Alphabet: In this study, two different English alphabets datasets are considered.

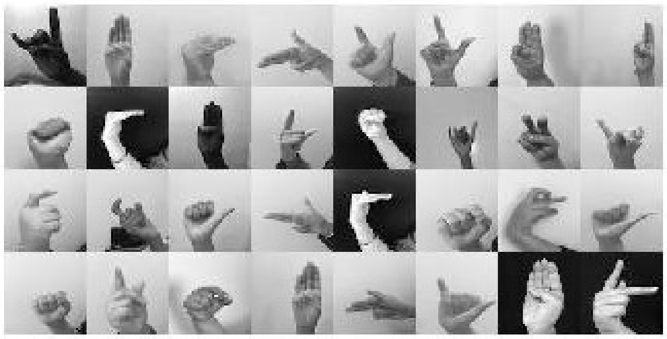

Fig. 8. Sample signs from the ArSL 2018 Arabic sign language dataset.

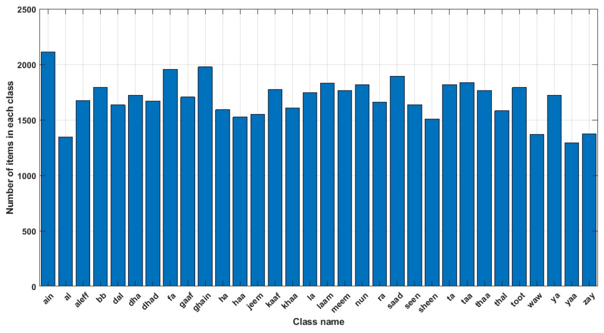

Fig. 9. Number of images in each class in ArSL dataset.

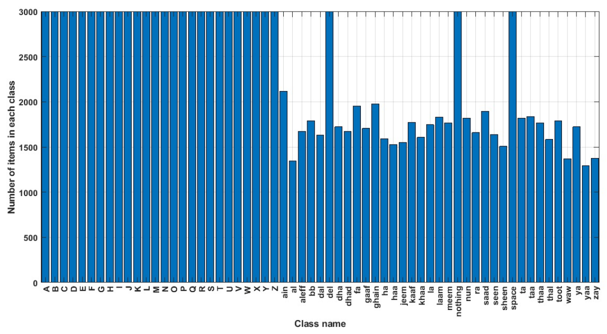

Fig. 10. Number of images in each class in the Arabic-English dataset.

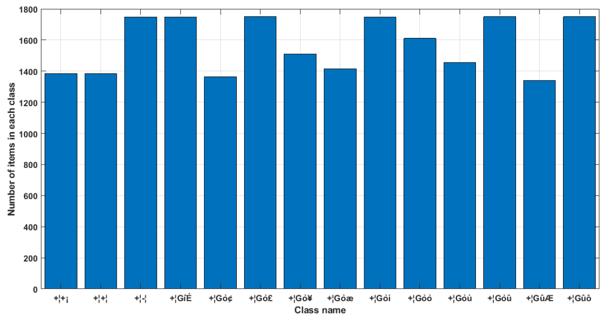

Fig. 11. Number of images in each class in the KSL dataset.

a) Kaggle dataset (ASL 2018): The 2018 update of the American Sign Language (ASL) alphabet dataset, available on Kaggle's data science repository, contains 29 classes for alphabet characters. Each class contains 3000 images captured with the intent to make a dataset for each character [37]. The dataset is divided into 70% for training, and the remainder is divided equally between validation and testing. Table IV shows the number of images for each partition. The data is rearranged randomly for each experiment.

b) Massey dataset (ASL 2012): The last version of the Massey dataset, introduced in January 2012, contains 1815 images of 26 ASL alphabet gestures [38]. Each class contains 70 images, so the data is balanced. The dataset is divided into 85% for training, and the remainder is divided equally between validation and testing, as shown in Table IV. The data is rearranged randomly for each experiment.

3) Arabic-English dataset: The Arabic-English dataset combines the ArSL 2018 and ASL 2018 datasets, and the model deals with them as one dataset. The dataset contains 61 classes; each class has a different number of images, dimensions, and colors. Fig. 10 shows the number of images in each class. The combined dataset is divided into 70% for training, and the remainder is divided equally between validation and testing, as shown in Table IV.

4) KSL consonant letters: Korean consonant letters are also available on Kaggle and were last updated in June 2021. It contains 14 classes for consonant Korean alphabet characters [39]. The data is supported by 21962 images for training and validation and 3790 for the test. The images in the Korean dataset are imbalanced distributed, as shown in Fig. 11. Training data is divided into 85% for training and 15% for validation, as shown in Table IV. Data is randomly fed to the network at each iteration.

TABLE IV. DATA DIVISION BETWEEN TRAINING AND TESTING

Dataset	Total images	training images	val images	test images
ArSL	54049	37835	8110	8114
ASL2018	87000	60900	13050	13050
ASL2012	1815	1555	130	130
Arabic-English	141049	98735	21150	21164
KSL	25752	18671	3291	3790

B. Experimental Setup

The experiment was conducted using MATLAB® 2020 Deep Learning Toolbox running on a 2.60 GHz Intel i5 CPU with 8 GB RAM, Intel 4 K graphics, and AMD Radeon 7500M/7600M series.

To evaluate the robustness of the model against randomization and avoid bias towards certain parameters, the experiment was conducted ten times; in each trial, the dataset was randomly divided into training and testing, and the results were averaged [15]. Also, the dropout layer is added to avoid the overfitting problems [40]. The effectiveness of the proposed model is evaluated on five datasets: (1) the ArSL 2018 dataset; (2) the ASL 2018 dataset; (3) the ASL 2012 dataset; (4) the ArSL-ASL combination; and (5) the KSL 2021 dataset. The accuracy defined as Eq. 9 is used to evaluate the classification performance, where FC and TC denote the total number of

TABLE V. Classification Accuracy, Training and Test Time

Dataset	T_r Acc(%)	T_s Acc(%)	T_r time(h)	T_s time(ms)	std for T_s
ArSL 2018	100	98.7799	3.93	2.6698	0.1436
ASL 2018	100	100.00	6.46	3.1746	0.1433
ASL 2012	100	100.00	0.2558	1.4912	0.4876
ArSL-ASL	100	99.4897	10.5	3.0440	0.4066
KSL 2021	100	100.00	2.60	2.7319	0.0737

where T_r means training and T_s means test

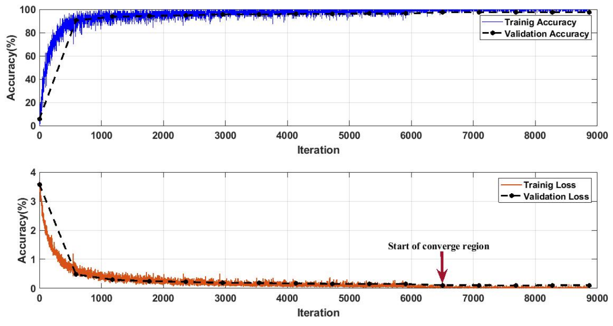

Fig. 12. Training and validation accuracy for ArSL.

false and correct instances of the test. Also, the sensitivity for each class is expressed as Eq. 10 and is used to express the evaluation of the performance for each class individually, where TC_c and TF_c are the numbers of correct and false instances in each class.

$$Accuracy = \frac{TC}{FC + TC} * 100 \tag{9}$$

$$TC_c = \frac{TC_c}{FC_c + TC_c} * 100 \tag{10}$$

C. Performance Evaluation of the Proposed SL-CNN Model

The proposed model was trained for 15 or 20 epochs, and the proposed algorithm achieves high accuracy at epoch number 15. Table V summarizes the training accuracy, test accuracy, training time, test time, and standard deviation for test time with 15 epochs for the proposed SL-CNN model on ArSL 2018, ASL 2018, ASL 2012, the combination between them (ArSL-ASL) datasets, and 20 epochs for KSL 2021. Fig. 12 to 16 show the training and test accuracies and the start of the loss function to converge for ArSL 2018, ASL 2018, ASL 2012, ArSL-ASL, and KSL, respectively. It is noted from the table that the training accuracy for all datasets is 100%, and the test accuracy is 100% for ASL 2018, ASL 2012, and KSL, as well as 98.8% and 99.5% for ArSL and the combined dataset. In addition, the training and test times are different from one dataset to another. Figs. 17, 18, 19, and 20 show the confusion matrices for ArSL, ASL 2018, ASL 2012, and KSL. The diagonal of the confusion matrix refers to the number of the correct element obtained by the model, while off-diagonal predictions are false.

D. Comparison with Other Models

In this section, we make a comparison between the SL-CNN model and other models published in other articles. The

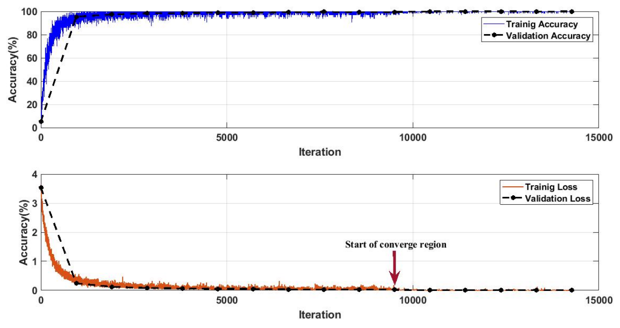

Fig. 13. Training and validation accuracy for ASL 2018.

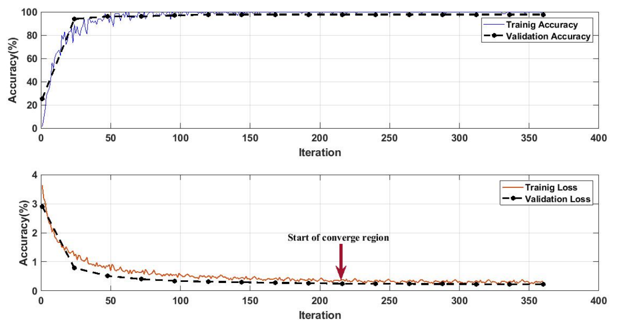

Fig. 14. Training and validation accuracy for ASL 2012.

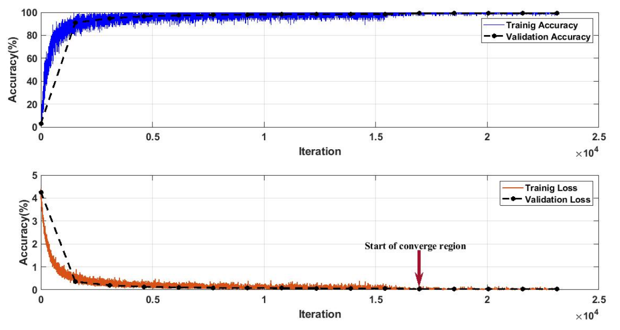

Fig. 15. Training and validation accuracy for ArSL-ASL.

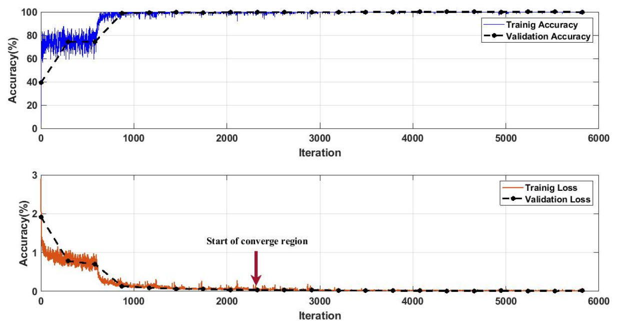

Fig. 16. Training and validation accuracy for KSL.

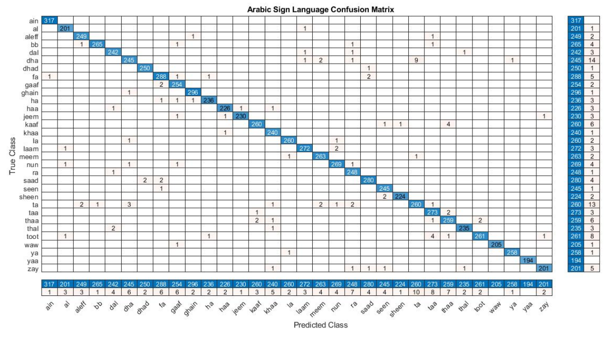

Fig. 17. Confusion matrix for ArSL.

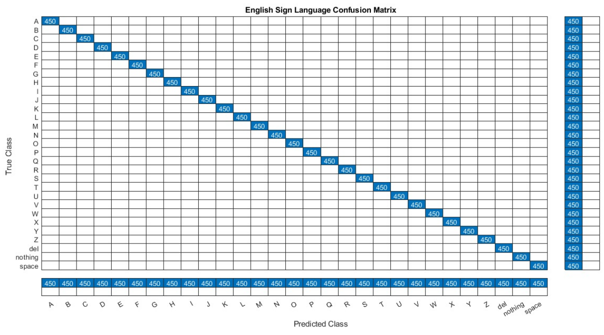

Fig. 18. Confusion matrix for ASL 2018.

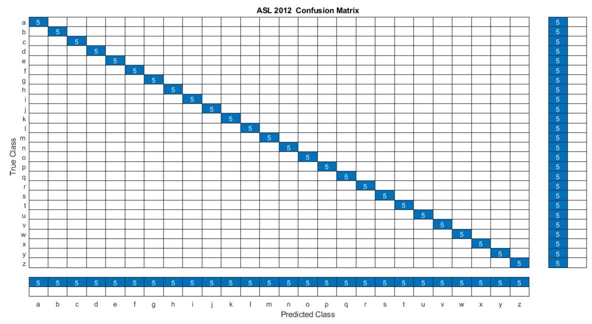

Fig. 19. Confusion matrix for ASL 2012.

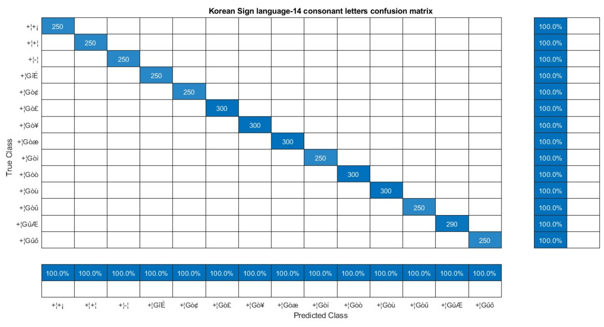

Fig. 20. Confusion matrix for KSL.

TABLE VI. Comparison of the Results between the Proposed Model and Other Previous Articles for ArSL

Author	Method	Dataset	Samples	Acc(%)
Latif, et al. [41]	CNN	ArSL 2018	54049	97.6
Elsayed, et al. [16]	CNN	ArSL 2018	54049	88.89
Alani,, et al. [15]	CNN	ArSL 2018	54049	97.29
proposed SL-CNN	CNN	ArSL 2018	54049	98.47
M. Zakariah, et al. [42]	EfficientNetB4	ArSL 2018	54049	95.0
Rehab, et al. [43]	VGGNet	ArSL 2018	54049	97.0

comparison is divided into three parts ArSL, ASL, and KSL. It is noted that the proposed SL-CNN model outperforms its peers because each parameter of the CNN model structure has been carefully selected, as mentioned in Section III, and doesn't depend on preprocessing techniques to improve the accuracy of the application like others; but it works on raw data. For the ArSL 2018 dataset, the proposed SL-CNN model achieves 98.47% accuracy, outperforming other CNN models proposed in the literature, e.g. [16], [41] and [15] as shown in Table VI. It should be noted that the proposed model uses raw data without any preprocessing. It should be highlighted that the models by [42] and [43] employed modern classification approaches (Transfer learning) but obtained poor accuracy when compared to the proposed SL-CNN model.

For the ASL, the proposed model is compared to the models proposed in Kaggle datasets [[8], [12], [18]] and [[12], [44], [45]] on the Massey dataset, which is shown in Table VII. The result of other state-of-the-art in literature [17] and [13] on collected but not publicly available datasets are also reported in Table VII. The CNN models proposed in [[46], [19]] achieved higher accuracy on selected images from the MNIST dataset. However, the proposed SL-CNN model is applied to all images in the recent Kaggle dataset, and the two versions of Massey beat all the previously suggested CNN models. Interestingly, the proposed model not only outperforms published models on ASL 2018 and ASL 2012 but also achieves 100% accuracy, in accordance with the recent results published on Kaggle [37]. It should be noted that the model by [47] achieved also 100% accuracy, however, on the Massey dataset 2011. Also, the model proposed in [13] got 99.3 % accuracy for ASL classification, but they used a collected dataset consisting of 240 images only.

Finally, for the Korean sign language, state-of-the-art methods in the literature are performed on collected but not publicly available datasets, as reported in VIII. The proposed model achieved 100% test accuracy for consonant characters, compared to the models by [20] who achieved 92.2% for vowel and consonant characters, and [21] who achieved 99.31% and 99.97% for vowel and consonant characters, respectively.

VI. Conclusion and Future Work

This paper proposes an approach to designing and analyzing an efficient CNN model for sign language recognition named SL-CNN based on a detailed study of CNN hyper-parameters, i.e., kernel size, number of layers, and filtering number in each layer. The performance of the proposed model was investigated for four sign language datasets (ArSL 2018, ASL 2018, ASL 2012, and KSL 2021), and one dataset was

TABLE VII. Comparison of the Results between the Proposed Model and Other Previous Articles for ASL

Author	Method	Acc (%)	Dataset	Samples
Aly, et al. [17]	PCANet and SVM	88.7	collected	60000
Raheem , et al. [13]	CNN	99.3	collected	240
Taskiran, et al. [47]	CNN	100	Massey 2011	900
A. Mannan, et al.[46]	CNN	99.67	MNIST	35000
C. Can, et al. [19]	CNN	99.91	MNIST	35000
Shin, et al. [12]	SVM	99.39	Massey 2012	1815
Rastgoo, et al. [44]	CNN	99.31	Massey 2012	1815
Rahman, et al. [45]	CNN	99.95	Massey 2012	1815
Proposed model	CNN	100	Massey 2012	1815
Shin, et al. [12]	SVM	87.6	ASL 2018	87000
Wardana, et al. [18]	CNN	99.81	ASL 2018	87000
Fierro, et al. [8]	CNN	96	ASL 2018	87000
Proposed model	CNN	100	ASL 2018	87000

TABLE VIII. Comparison of the Results between the Proposed Model and Other Previous Articles for KSL

Author	Method	Dataset	Classes	Acc (%)
[20]	SVM	collected	31 (14 $consonants$, 10$vowels$, and 7$double$ $vowels$)	92.2% (both)
[21]	$Gaussian$ $Modeling$ and $Likelihood$ $Estimation$	collected	12 (6$consonants$, 6$vowels$)	99.31% ($vowel$) 99.97% ($consonant$)
Proposed	CNN	Kaggle	14 (consonant)	100% (consonant)

obtained as a merge of the first two datasets (ArSL and ASL 2018). Table VIII shows the comparison of the results between the proposed model and other previous articles for KSL. The results show that the proposed SL-CNN model outperforms the other models in terms of classification accuracy for all datasets. In summary, the proposed model achieved accuracy for 98.8%, 100%, 99.5%, 100% and 100% for ArSL2018, ASL 2018, combining both, ASL 2012 and KSL 2021, respectively without resorting to any data preprocessing. As suggestions for future work, it is interesting to investigate the performance of the proposed model on sign language datasets consisting of words and sentences. Another possibility is to implement the proposed model in, e.g., an educational system for people who suffer from hearing problems.

Acknowledgment

Special thanks to Dr. Ahmed Mahmoud Alenany and Dr. Mohamed L. Elsayed for the English review of this manuscript.

References

[1] E. McPhillips, "World wide hearing loss: Stats from around the world," *Audicus. Retrieved September*, vol. 10, p. 2022, 2022.

[2] M. Mustafa, "A study on arabic sign language recognition for differently abled using advanced machine learning classifiers," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 4101–4115, 2021.

[3] C. K. Lee, K. K. Ng, C.-H. Chen, H. C. Lau, S. Chung, and T. Tsoi, "American sign language recognition and training method with recurrent neural network," *Expert Systems with Applications*, vol. 167, p. 114403, 2021.

[4] S.-K. Ko, J. G. Son, and H. Jung, "Sign language recognition with recurrent neural network using human keypoint detection," in *Proceedings of the 2018 conference on research in adaptive and convergent systems*, 2018, pp. 326–328.

[5] R. Rastgoo, K. Kiani, and S. Escalera, "Sign language recognition: A deep survey," *Expert Systems with Applications*, vol. 164, p. 113794, 2021.

[6] E.-S. M. El-Alfy and H. Luqman, "A comprehensive survey and taxonomy of sign language research," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105198, 2022.

[7] A. Al-Naji, K. Gibson, S.-H. Lee, and J. Chahl, "Real time apnoea monitoring of children using the microsoft kinect sensor: a pilot study," *Sensors*, vol. 17, no. 2, p. 286, 2017.

[8] K. R. Perez-Daniel and A. N. Fierro Radilla, "Siamese convolutional neural network for asl alphabet recognition," *OPENAIRE*, 2020.

[9] R. Alzohairi, R. Alghonaim, W. Alshehri, and S. Aloqeely, "Image based arabic sign language recognition system," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 3, 2018.

[10] M. Abdo, A. Hamdy, S. Salem, and E. M. Saad, "Arabic alphabet and numbers sign language recognition," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 11, pp. 209–214, 2015.

[11] C. Dong, M. C. Leu, and Z. Yin, "American sign language alphabet recognition using microsoft kinect," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 44–52.

[12] J. Shin, A. Matsuoka, M. A. M. Hasan, and A. Y. Srizon, "American sign language alphabet recognition by extracting feature from hand pose estimation," *Sensors*, vol. 21, no. 17, p. 5856, 2021.

[13] A. A. Abdulhussein and F. A. Raheem, "Hand gesture recognition of static letters american sign language (asl) using deep learning," *Engineering and Technology Journal*, vol. 38, no. 6, pp. 926–937, 2020.

[14] N. Hasasneh, A. Hasasneh, and S. Taqatqa, "Towards arabic alphabet and numbers sign language recognition," 2017.

[15] A. A. Alani and G. Cosma, "Arsl-cnn: a convolutional neural network for arabic sign language gesture recognition," *Indonesian journal of electrical engineering and computer science*, vol. 22, 2021.

[16] E. K. Elsayed and D. R. Fathy, "Sign language semantic translation system using ontology and deep learning," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, 2020.

[17] W. Aly, S. Aly, and S. Almotairi, "User-independent american sign language alphabet recognition based on depth image and pcanet features," *IEEE Access*, vol. 7, pp. 123 138–123 150, 2019.

[18] B. K. Wardana, E. Rachmawati, and T. A. B. Wirayuda, "Pengenalan gestur tangan statis menggunakan cnn dengan arsitektur efficient-net b4," *eProceedings of Engineering*, vol. 8, no. 2, 2021.

[19] C. Can, Y. Kaya, and F. Kılıç, "A deep convolutional neural network model for hand gesture recognition in 2d near-infrared images," *Biomedical Physics & Engineering Express*, vol. 7, no. 5, p. 055005, 2021.

[20] Y. Na, H. Yang, and J. Woo, "Classification of the korean sign language alphabet using an accelerometer with a support vector machine," *Journal of Sensors*, vol. 2021, pp. 1–10, 2021.

[21] I. Yeo and H.-C. Shin, "Novel korean finger language recognition using emg and motion sensors," in *2018 International Conference on Information Networking (ICOIN)*. IEEE, 2018, pp. 837–839.

[22] A. H. Barshooi and A. Amirkhani, "A novel data augmentation based on gabor filter and convolutional deep learning for improving the classification of covid-19 chest x-ray images," *Biomedical Signal Processing and Control*, vol. 72, p. 103326, 2022.

[23] D. A. Pitaloka, A. Wulandari, T. Basaruddin, and D. Y. Liliana, "Enhancing cnn with preprocessing stage in automatic emotion recognition," *Procedia computer science*, vol. 116, pp. 523–529, 2017.

[24] H. Moujahid, B. Cherradi, O. El Gannour, W. Nagmeldin, A. Abdelmaboud, M. Al-Sarem, L. Bahatti, F. Saeed, and M. Hadwan, "A novel explainable cnn model for screening covid-19 on x-ray images," *Comput. Syst. Sci. Eng.*, vol. 46, no. 2, pp. 1789–1809, 2023.

[25] A. Amirkhani, A. H. Barshooi, and A. Ebrahimi, "Enhancing the robustness of visual object tracking via style transfer." *Computers, Materials & Continua*, vol. 70, no. 1, 2022.

[26] R. Garg, S. Maheshwari, and A. Shukla, "Decision support system for detection and classification of skin cancer using cnn," in *Innovations in Computational Intelligence and Computer Vision: Proceedings of ICICV 2020*. Springer, 2021, pp. 578–586.

[27] S. Wen, J. Chen, Y. Wu, Z. Yan, Y. Cao, Y. Yang, and T. Huang, "Ckfo: Convolution kernel first operated algorithm with applications in memristor-based convolutional neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 8, pp. 1640–1647, 2020.

[28] F. F. Li, J. Justin, and S. Yeung, "Course notes on cs231n: Convolutional neural networks for visual recognition," 2018.

[29] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[30] M. Coletti, D. Lunga, J. K. Bassett, and A. Rose, "Evolving larger convolutional layer kernel sizes for a settlement detection deep-learner on summit," in *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*. IEEE, 2019, pp. 36–44.

[31] W. S. Ahmed *et al.*, "The impact of filter size and number of filters on classification accuracy in cnn," in *2020 International conference on computer science and software engineering (CSASE)*. IEEE, 2020, pp. 88–93.

[32] J. Brownlee, "A gentle introduction to padding and stride for convolutional neural networks," *Machine Learning Mastery*, 2019.

[33] S. Shin, Y. Lee, M. Kim, J. Park, S. Lee, and K. Min, "Deep neural network model with bayesian hyperparameter optimization for prediction of nox at transient conditions in a diesel engine," *Engineering Applications of Artificial Intelligence*, vol. 94, p. 103761, 2020.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[35] K. Alkharabsheh, S. Alawadi, V. R. Kebande, Y. Crespo, M. Fernández-Delgado, and J. A. Taboada, "A comparison of machine learning algorithms on design smell detection using balanced and imbalanced dataset: A study of god class," *Information and Software Technology*, vol. 143, p. 106736, 2022.

[36] L. Ghazanfar, A. Jaafar, M. Nazeeruddin, A. Roaa, and A. Rawan, "Arabic alphabets sign language dataset (arasl)," *Mendeley Data*, vol. 1, p. 2018, 2018.

[37] AKASH. Asl alphabet. [Online]. Available: https://www.kaggle.com/grassknoted/asl-alphabet

[38] A. Barczak, N. Reyes, M. Abastillas, A. Piccio, and T. Susnjak, "A new 2d static hand gesture colour image dataset for asl gestures," 2011.

[39] BIO. Korean sign language-14 consonant letters. [Online]. Available: https://www.kaggle.com/ttr9904ewhaincom/korean-sign-language14-consonant-letters/activity

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[41] G. Latif, N. Mohammad, R. AlKhalaf, R. AlKhalaf, J. Alghazo, and M. Khan, "An automatic arabic sign language recognition system based on deep cnn: an assistive system for the deaf and hard of hearing," *International Journal of Computing and Digital Systems*, vol. 9, no. 4, pp. 715–724, 2020.

[42] M. Zakariah, Y. A. Alotaibi, D. Koundal, Y. Guo, M. Mamun Elahi *et al.*, "Sign language recognition for arabic alphabets using transfer learning technique," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[43] R. M. Duwairi and Z. A. Halloush, "Automatic recognition of arabic alphabets sign language using deep learning." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 12, no. 3, 2022.

[44] R. Rastgoo, K. Kiani, and S. Escalera, "Multi-modal deep hand sign language recognition in still images using restricted boltzmann machine," *Entropy*, vol. 20, no. 11, p. 809, 2018.

[45] M. M. Rahman, M. S. Islam, M. H. Rahman, R. Sassi, M. W. Rivolta, and M. Aktaruzzaman, "A new benchmark on american sign language recognition using convolutional neural network," in *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*. IEEE, 2019, pp. 1–6.

[46] A. Mannan, A. Abbasi, A. R. Javed, A. Ahsan, T. R. Gadekallu, Q. Xin *et al.*, "Hypertuned deep convolutional neural network for sign language recognition," *Computational intelligence and neuroscience*, vol. 2022, 2022.

[47] M. Taskiran, M. Killioglu, and N. Kahraman, "A real-time system for recognition of american sign language by using deep learning," in *2018 41st international conference on telecommunications and signal processing (TSP)*. IEEE, 2018, pp. 1–5.