IIUC Journal of Science and Engineering
Vol.-1, Issue-1, December 2023, pp. 183-200
ISSN: 3005-5873

Gesture recognition through binary video transformation with IoT

Muhammed Nazmul Arefin
Department of Computer Science and Engineering
International Islamic University Chittagong (IIUC), Bangladesh.

Article info

Keywords
ASL
CNN
IoT
Neural Network

Abstract

Sign language is used by those who are deaf to communicate. The use of American Sign Language (ASL) is widespread all over the world. For those who are deaf, it is problematic because many others do not understand sign language. Both deaf individuals and any service staff would greatly benefit if there was an innovative solution that could identify signs and convert them into simple English. Convolutional neural networks (CNN), one of the subtypes of neural networks, are utilized for picture applications and have a wide range of uses. For video applications with time components, we use Recurrent Neural Networks (RNN) or Long Short-Term Memory (LSTM) networks. However, these networks require a substantial amount of conditioning before producing accurate results, leading to longer processing times. In this work, we achieved improved results with 99.40% accuracy for alphabets and 99.70% accuracy for words when using CNN to recognize gestures from video.

Corresponding author
Muhammed Nazmul Arefin
Department of Computer Science and Engineering, International Islamic University Chittagong, Chattogram, Bangladesh - 4318. Email: nazmul.arefin@iiuc.ac.bd

1. Introduction

According to the World Health Organization, there are currently 466 million people who are deaf. It accounts for approximately 5% of the world's total population. Yet the people who know sign language are rare. Though more and more people are learning sign language gradually, it is not the norm. So, in many instances, it might happen that a deaf person knows sign language but the person with whom he is trying to communicate is not aware of any sign language. This limits the scope of communication for deaf people in all areas of life. Due to this, many technological advancements are being used to ease this problem. Some computer software technologies can recognize sign language through the use of a camera. There is also some specialized

hardware that can be used by deaf people to communicate with people who don't know any sign language. In this research, we will use a computer vision technique to recognize American Sign Language (ASL) alphabets or words and translate them into readable words, which will greatly help deaf people around the globe.

Traditionally, the most common approaches in the field of video applications involving temporal dynamics, including gesture detection, have relied on Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks. For the purpose of recording sequential patterns across time, these designs are ideal. But their effectiveness depends on thorough conditioning and preprocessing, which can greatly increase processing time. In our recent research endeavor, we sought to explore alternative methodologies that could potentially mitigate the processing overhead while maintaining or even enhancing accuracy levels. Our investigation led us to experiment with Convolutional Neural Networks (CNNs), a class of deep learning models primarily associated with image processing tasks. While CNNs have traditionally been applied to static image analysis, their ability to capture spatial hierarchies and patterns made them a compelling candidate for temporal data analysis as well. Surprisingly, our findings revealed that CNNs exhibited remarkable performance when tasked with recognizing gestures from video data. Notably, we achieved an impressive accuracy rate of 99.40% for alphabet recognition and an even higher accuracy of 99.70% for word recognition. These results surpassed the performance of traditional RNN and LSTM approaches, while also mitigating the need for extensive preprocessing steps.

2. Literature review

In most of the cases, camera equipment used to get image data and analysis of those images has been done based on popular analysis methods to get the gesture information (Luzanin & Plancak, 2014). As the camera is very much available in all devices and not much high price which gives much opportunity of human machine interaction (Chen et al., 2017; Lu, Chen, Li, Zhang, & Zhou, 2014; Sun et al., 2018). Another significant method we found in some research works is Hidden Markov Model. Hidden state and actual state are the two states of the Hidden Markov Model (Liu, Chen, Jafari, & Kehtarnavaz, 2014; Liao et al., 2017). Zhang and Deng (2017) proposed a method called dynamic gesture track recognition strategy based on HMM-CART model and showed higher accuracy of dynamic gesture recognition. Wang, Xia, Cai, Gao, and Cattani (2012) trained the time-based

information-handling HMM using the Baum-Welch technique.

LeCun, Bengio, and Hinton (2015) provided a basic outline of the Neural Network types and their application about speech recognition, visual object recognition, drug discovery and genomics studies. They also discussed about the power of back propagation and how it enabled more sophisticated learning in terms of dynamic data set. They concluded that the advanced reasoning of the algorithm will be in charge of the majority of successful learning in the future. O'shea and Nash (2015) discussed about the basics of the Convolutional Neural Network which is under the field of Artificial Neural network also the different variety of neurons and the group of them. These groups are called layers which is discussed in the section of CNN. As mentioned, these CNN layers are convolutional layers, dense layers/fully connected layers, pooling layers etc. In their papers they also showed the patterns that have been learned from the data set which looks almost identical to the data set images. they used MNIST database to train the network to recognize hand written digits. Wenjin et al. (2018) used CNN to recognize the alphabets of the ASL. As they required huge amount of training data, they captured some and used data augmentation technique to multiply those data into large data set. Moreover, they used a technique called inference fusion which enabled them to increase the accuracy of the model. Masood, Srivastava, Thuwal, and Ahmad (2018) attempts to recognize words of the ASL. Words of the ASL are not static images so conventional CNN is not feasible here. Being temporal sequence of images gestures requires special NN architecture known as Recurrent Neural Network. The authors used both CNN and RNN to recognize the different characteristics of the video data. By designing appropriate model of the network, they achieved above 95 percent accuracy. Saleem (2020) trained a CNN to recognize 24 ASL alphabet and tried to finger spell any word as required. For the training data set they used both image intensity data and depth data. Though in our project we did not use color image for training, this paper showed that color images and depth info can be useful to better predict the desired classes. They also showed that their proposed model is 3% more accurate than the previous works. By using both the architecture geometric information is captured using CNN and time series information is captured using RNN. Their work was able to recognize 27 ASL gestures with 69.2% accuracy. They collaborated with computational ASL recognition researcher to create this new data set. They used Kinect 2.0 sensor to record the gestures. Ko, Son, and Jung (2018) proposed a method that uses human key point vector to recognize Korean sign language gestures. This key point vector has different angles and deviation for different signs. These values are then fed into the

RNN for training. They introduced KETI sign language dataset containing 10,480 videos. Their system showed 89.5% accuracy even when there is insufficient training data. Yu, Si, Hu, and Zhang (2019) reviewed Recurrent Neural Network and several Long Short-Term Memory (LSTM) cell architectures. They pointed out the drawback of the RNN and discussed the importance of LSTM for different application. The authors summarized several types of LSTM networks and their possible application. In the final remarks they pointed out some techniques to further improve RNNs effectiveness. Deshmukh and Jagade (2022) proposed the CNN Algorithm to detect Hand Sign Language. The neighborhood framework or the webcam camera casing are used as information in this suggested framework. The convolution neural network algorithm-based classifiers receive input images that have been handled. It organizes the image and turns it into a model. Finally, the desired result comes to fruition. Atitallah et al. (2021) suggested using a low-complexity Electrical Impedance Tomography imaging system for hand sign detection and monitoring. They also incorporated a virtual hand model for visualization, a Gauss-Newton image reconstruction technique, a robust CNN based hand sign classification, and an 8- electrode electronic interface. Following table includes the comparative analysis of ASL.

Table I
Comparative analysis

Study	Approach	Key Findings
Chen et al. (2017)	Interactive image segmentation in hand gesture recognition	Proposed an interactive image segmentation method for hand gesture recognition
		Achieved accurate hand segmentation by combining color, depth, and skin detection
Liu, Chen, Jafari, and Kehtarnavaz (2014)	Fusion of inertial and depth sensor data	Investigated the fusion of inertial and depth sensor data for robust hand gesture recognition
		Combined sensor information to improve recognition accuracy and robustness
LeCun, Bengio, and Hinton (2015)	Deep learning	Introduced deep learning as a powerful approach for various machine learning tasks
		Highlighted the use of convolutional neural networks (CNNs) for image analysis

O'Shea and Nash (2015)	Introduction to convolutional neural networks	Provided an introduction to convolutional neural networks (CNNs)
		Discussed the architecture and training of CNNs for image recognition
Tao, Leu, & Yin (2018)	American Sign Language (ASL) recognition using CNN	Developed a CNN-based framework for ASL alphabet recognition
		Utilized Multiview augmentation and inference fusion to enhance recognition accuracy
Liao et al. (2017)	Simultaneous calibration of multiple Kinect and cameras	Proposed a joint optimization approach for simultaneous calibration of multiple Kinect and external cameras
		Achieved accurate calibration and alignment of multiple sensors
Lu, Chen, Li, X Zhang, and Zhou (2014)	Hand gesture recognition framework and wearable interaction prototype	Presented a hand gesture recognition framework and prototype for wearable devices
		Integrated vision-based hand gesture recognition with mobile devices
Luzanin and Plancak (2014)	Hand gesture recognition using low-budget data glove	Used a low-budget data glove and cluster-trained probabilistic neural network for hand gesture recognition
		Demonstrated the effectiveness of the approach in recognizing hand gestures
Masood, Srivastava, Thuwal, and Ahmad (2018)	Real-time sign language gesture recognition using CNN and RNN	Developed a real-time system for sign language gesture recognition
		Utilized CNN and RNN architectures to recognize and translate sign language gestures
Sun, et al. (2018)	Gesture recognition based on Kinect and sEMG fusion	Proposed a fusion-based approach combining Kinect and surface electromyography (sEMG) signals for gesture recognition
		Achieved improved recognition accuracy by leveraging multiple modalities
Wang, Xia, Cai, Gao, and Cattani (2012)	Hidden-Markov-Models-Based dynamic hand gesture recognition	Employed Hidden Markov Models (HMMs) for dynamic hand gesture recognition
		Captured temporal dynamics in hand gestures using HMMs
Zhang and Deng (2017)	Dynamic gesture recognition based on LeapMotion and HMM-CART	Utilized LeapMotion sensor data and a hybrid HMM-CART model for dynamic gesture recognition
		Achieved accurate recognition of dynamic hand gestures

Our main objective in this research is to use state-of-the-art techniques of Artificial Intelligence (AI) and create a program that will recognize a portion of the vocabulary of ASL. For instance, our program will recognize various predetermined alphabets from the captured data set from the camera. To do that, first we will create a data set of the alphabets and selected words. Then we will make our program learn the patterns of the words without remembering all the unimportant details. In order to make our computer learn the words, we have to train our program using the aforementioned data set. In computer vision, 'learning' is facilitated by the technology called Neural Network (NN). NN is currently very popular among Machine Learning (ML) researchers. We also need to make some changes to the collected images from the camera because NNs do not work with the raw image data. After doing these necessary steps, our system will be able to recognize many static images as alphabets and moving images as words of the English language.

In this research, we will create a program that will recognize 24 letters of the English alphabet without 'J' and 'Z' as well as 50 English words using American Sign Language. Since ASL words are an example of temporal sequence data, we will transform this data in such a way that we can use CNN instead of RNN to train our program.

To accomplish our goal, we will follow a step-by-step approach. Firstly, we will create a standard image dataset for the individual letters involved in the American Sign Language (ASL). Next, we need to find a way to transform the temporal sequence of data, typically found in video applications, into static image data that can be processed by a Convolutional Neural Network (CNN). Using this transformed static data, we will create a separate dataset specifically for recognizing words in ASL. Prior to inputting the images into the CNN, we will preprocess them to enhance their quality and optimize the learning process. Furthermore, we will design two separate CNN architectures: one for recognizing individual letters and another for recognizing complete words. These models will be trained extensively using the constructed datasets to ensure optimal performance. Finally, to assess the accuracy and generalization capabilities of the models, we will test them using a set of images that are outside the original dataset. This comprehensive approach will enable us to effectively recognize both individual letters and words in ASL.

3. Methodology

Our research is an example of multiclass classification in AI. Convolutional neural networks (CNN) are used to create the system's fundamental deep

learning model. The data set is created by us for both alphabets and words. The image set for words is different from the conventional images of CNN because ASL words are hand gestures, which are a temporal sequence of data. For this reason, we have converted time series data into static image data first, and then those images will eventually create the data set for the ASL words. Then we have compared both CNN model architectures for words and alphabets.

A. Multiclass Classification

Multiclass classification involves categorizing data into multiple categories, while binary classification focuses on distinguishing between two categories. For our research, we will be solving a multiclass classification problem because our designed CNN architecture will detect the 24 alphabets of the English language using ASL signs. Another model will be trained to classify 50 words of the ASL containing gesture data. Below is a visual example of a multiclass classification problem where the problem domain contains three groups.

B. Dataset pre-processing

As we have selected CNN as our ML algorithm, we preprocessed the data as required for the CNN. At first, we collected still images from the camera for the alphabets of the ASL. Then we strategically converted the video data of the gesture into still image data for the words of the ASL. After that we organized the collected data into different folders to indicate the different classes.

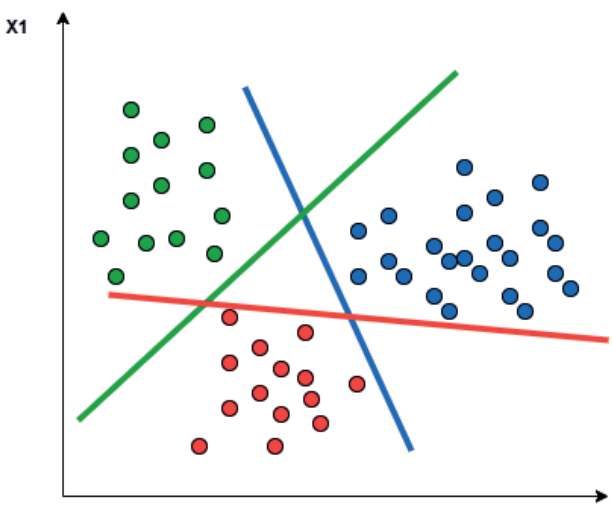

Figure 1
Multiclass classification

i. *Data collection*

For alphabetical data: We began capturing raw camera frames for alphabetical data. In the raw footage, we drew a green rectangle in the middle where we placed our hand and did the required pose. That image was then captured, and that captured image was one of the many images that composed the final data set. An example of the raw frame and captured image is shown below. It is the sign for the alphabet 'W'.

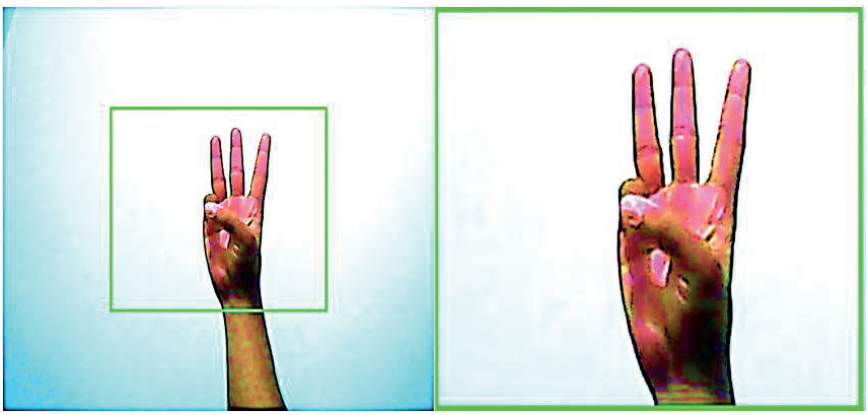

Figure 2
Raw image and cropped image

We took many photos for an alphabet and stored them into a folder. Since this image capture process is lengthy, we automated this task for each alphabet. At first, the camera starts showing the raw frames from the camera. Then the user places the hand in the rectangle. After a previously set time, a camera starts recording every frame and converting the frame into an image. Then every image is stored into that folder until the required number of images have been saved. For other alphabets, the same process is followed, and images are saved into another directory.

For word data: The signs for the ASL words are not static but moving gestures of hands and fingers. We created a way to convert this time series data into a static still image. We stood in front of the camera and did gestures for a word. With the help of a computer vision library, we captured only the movement of the user. The built-in function of the library '*BackgroundSubtractor*' captures only the movement in a video and shows this movement in the form of a binary image.

For a gesture, we collected all the frames of the binary image during a gesture and then added all of them. This added binary image is now the testament of the whole movement, but in the form of a still image. We

repeated a gesture in front of the camera 500 times and captured 500 images. These 500 images will form a dataset for an ASL word.

ii. Preparation of the dataset

As mentioned, we collected images for both alphabets and words. For alphabets, we captured 2000 images for each of the 24 classes. And for words, we captured 500 images for every gesture class. We created different folders for each class and put the corresponding images in the folders to create separate datasets.

C. CNN model

The overall success of a convolutional neural network depends on its model architecture, which means the design and sequential structure of different layers within the CNN. In our model, we have tested many configurations and selected one that gave us satisfactory results. The selected models for alphabets and words are given below.

i. Model for alphabets

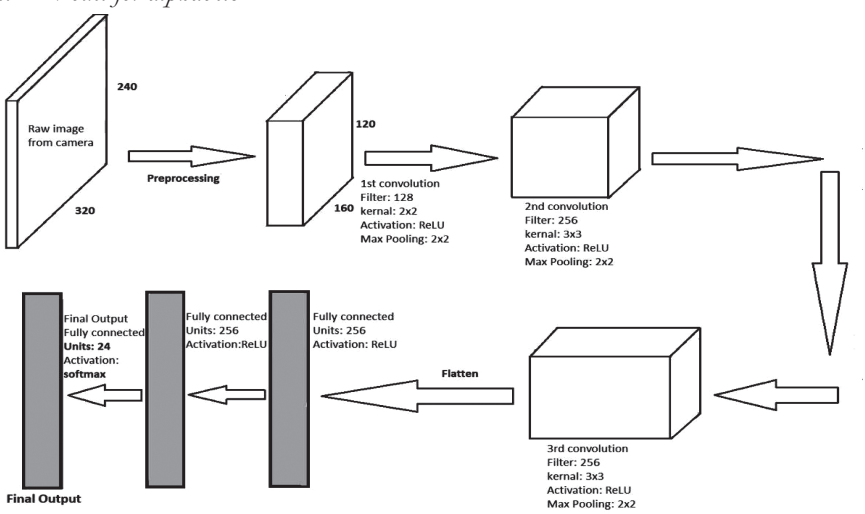

Figure 3
CNN model for alphabets

The CNN model for the alphabets uses 3 convolution layers, 3 pooling layers, and 3 fully connected layers. A schematic of the model is shown in the figure 3.

The first layer of our CNN alphabet is a 2D convolution layer, which does the spatial convolution over the images. In this layer, 128 filters are used as the dimensionality of the output space. According to the Keras

documentation, the first layer needs the dimension of the input images. So, we added input shapes from our data set. In this layer, we used 2 x 2 filter sizes to do the convolutions. To activate this layer, we used a rectified linear function, also known as ReLU. This function will output directly if the input is positive, and otherwise it will output zero. This activation function is the default go-to function because it makes the training easier and often leads to better performance. It also solves the vanishing gradient problem. The pooling layer down sample the input image and make it ready for the next convolution pass. We used the max pooling 2D from the Keras library, which selects the maximum value from neighboring pixels. Our selected pool size is 2 x 2. Here no stride length is mentioned, and by default it is set to 2 x 2 as the pooling size. In the second Conv2D layer, the number of filters and output space are the same as the 1st convolution layer. In the second layer, we don't have to mention the input shape like before. Right after the 2nd convolution layer, we used the same ReLU activation function. For the next pooling layer, we used MaxPooling2D as before with a 2 x 2 pooling size. In the third Conv2D layer, the number of filters and output space are the same as in the 1st convolution layer. In the second layer, we don't have to mention the input shape like before. Right after the 2nd convolution layer, we used the same ReLU activation function. For the next pooling layer, we used MaxPooling2D as before with a 2 x 2 pooling size. In this layer, all the previous input of a 2-dimensional matrix is flattened into a vector, which is fed to the fully connected layer next. In every CNN problem, a fully connected layer is crucial for generating prediction. In this dense layer, we used 256 neurons. For the dense layer, we used the same ReLU activation function. Another dense layer is used here with a 256-unit size. For the second dense layer, we used the same ReLU activation function. In the final output layer, we have used another fully connected layer. Since it is the last layer of the model, this layer has 24 neurons for each of the recognizable alphabets. For activation of the neurons, we used softmax because our problem domain is multiclass classification problems.

ii. Model for words

The model architecture to recognize the words consists of 2 convolution layers, 2 pooling layers, and 3 fully connected layers. The figure 4 shows the layers and neurons of the model. The first layer of our CNN for words is a 2D convolution layer, which does the spatial convolution over the images. In this layer, 256 filters are used as the dimensionality of the output space. The dimension of the input images is required by the first layer, according to the Keras documentation. So, we used our data set to add input shapes. In this

layer, we used 3 x 3 filter sizes to do the convolutions. After that, we followed every step from the previous model except the final output layer. In this final output layer, we have used another fully connected layer. Since it is the last layer of the model, this layer has 50 neurons for each of the recognizable words. For activation of the neurons, we used softmax just like before.

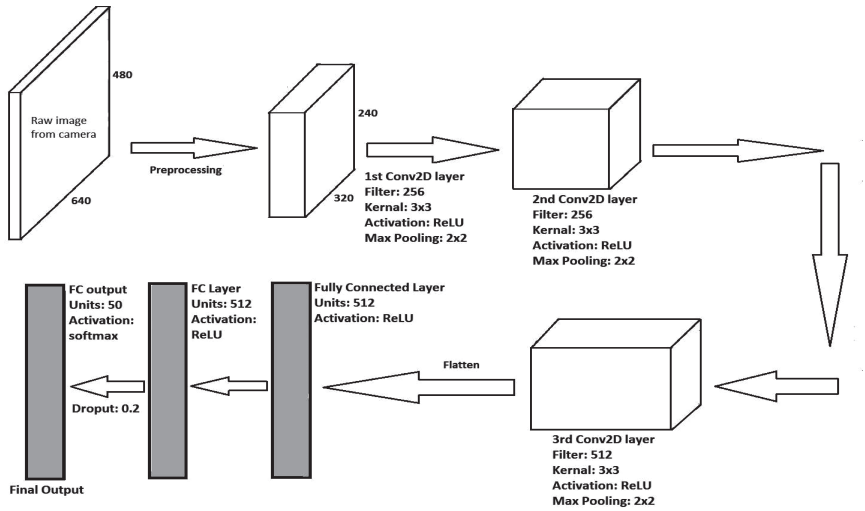

Figure 4
CNN model for words

D. Loss function for classification

Typically, neural networks try to minimize the error in every successive pass and update the weights of the neurons accordingly. There are numerous loss functions to be used from the Keras library. Binary Crossentropy, Categorical Crossentropy, Mean Absolute Error, Mean Squared Error, Sparse Categorical Crossentropy, Squared Hinge are some of the loss functions from the library. We used sparse categorical cross entropy as our loss function because our problem is a multiclass classification problem.

$$CCE(p,t) = -\sum_{c=1}^{c} t_{0,c}\log\left(p_{0,c}\right) \dots\dots\dots\dots \quad (1)$$

E. Optimization algorithm

An optimization algorithm acts like a support for the loss function. When compiling, any loss function may require a huge amount of time and resources, depending on the starting neuron weight values. With the use of an optimizer, we reduce the resource requirements in training. There are many optimizers in the Keras library, which we used for our training. Some

of them are RMSprop, SGD, ADAM, Adagrad, Ftrl etc. All of these optimizers have different use cases, and there is no default optimizer that works best. We experimented with these optimizers and chose the Adam optimizer because it has lower memory requirements and is suitable for numerous parameter handlings.

4. Result and discussion

As mentioned before, we have created our own data set from scratch. The data set for words contains a total of 25000 images, with 500 images in each folder for each class. All the images are binary images for words, which contain the movement information of the corresponding word. The resolution of the images is 640 x 480 and their average size is around 35 kilobytes. Before training, their size had been decreased to speed up the training process. Similar to alphabets, this data set was also divided into two parts; one for training and one for testing. 20,000 images were used for training, and 5,000 images will be used for testing. We utilized the Python programming language for this research. For the required libraries, programming language, software, and all other dependencies, we used the Spyder IDE of Anaconda as our distribution platform. For the libraries, we have used TensorFlow, OpenCV, Matplotlib, Numpy, OS and Keras. After successful training, we evaluated our designed architecture by using the Keras library. After 10 epochs, both the accuracy for alphabets and words were satisfactory, at more than 99%. Before finalizing our proposed NN architecture, we have experimented with many designs and parameters to make sure we get the desired accuracy. During the design of the architecture, it is accepted among researchers that there is no one design guideline to follow but rather trial and error with many structures. The variables we experimented with are-

- Number of CNN layers.
- Number of Neurons and filters within the CNN layers.
- Different activation function.
- Pooling size and strides.
- Number and size of dense layers.

For alphabets, we selected a model architecture that gave us satisfactory accuracy. We trained our CNN 10 times with the dataset, and each time the accuracy has increased. Every epoch took around 13 hours to complete. Line charts are given below to illustrate the learning of the CNN model.

Similar to alphabets, we chose a model architecture for word after many trials. We also trained the model for 10 epochs, and in every epoch the accuracy has

increased and the loss has decreased. Every category had 500 images, which is relatively low compared to the alphabets, which contained 2000 images per category.

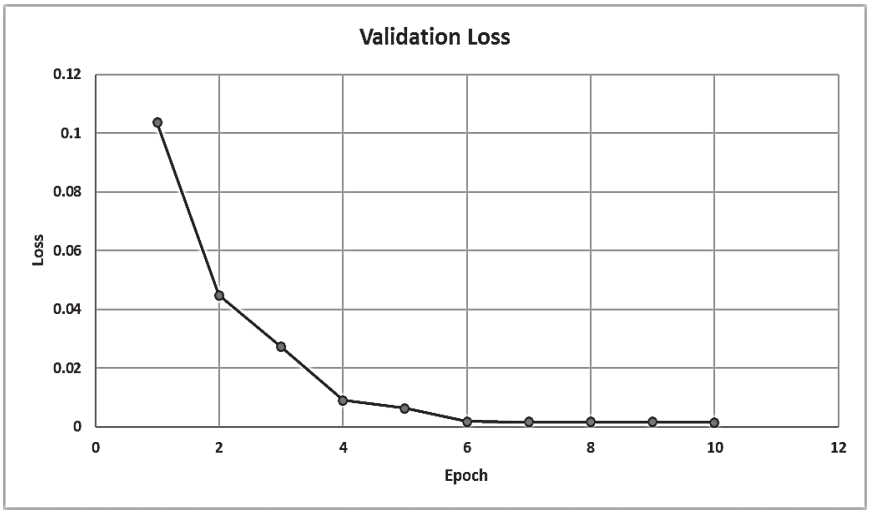

Figure 5
Line chart of the validation loss (Alphabet)

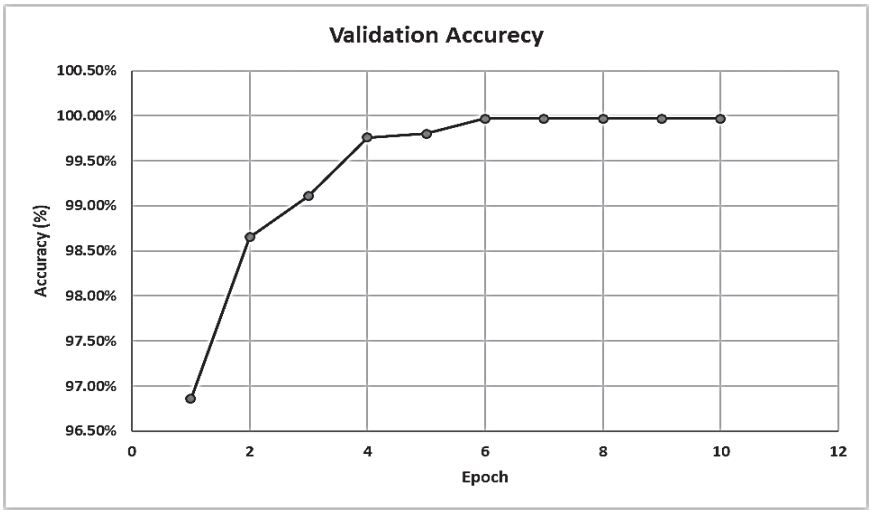

Figure 6
Line chart of the validation accuracy (Alphabet)

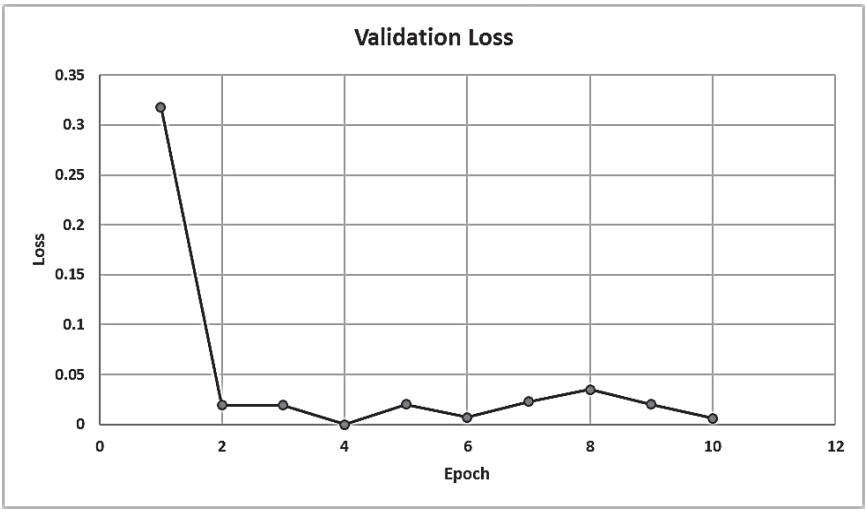

Figure 7
Line chart of the validation loss (Word)

As a result, training took less time than the alphabets. Moreover, our CNN architecture was simpler than the alphabet's model. Each epoch took around 2 hours to complete.

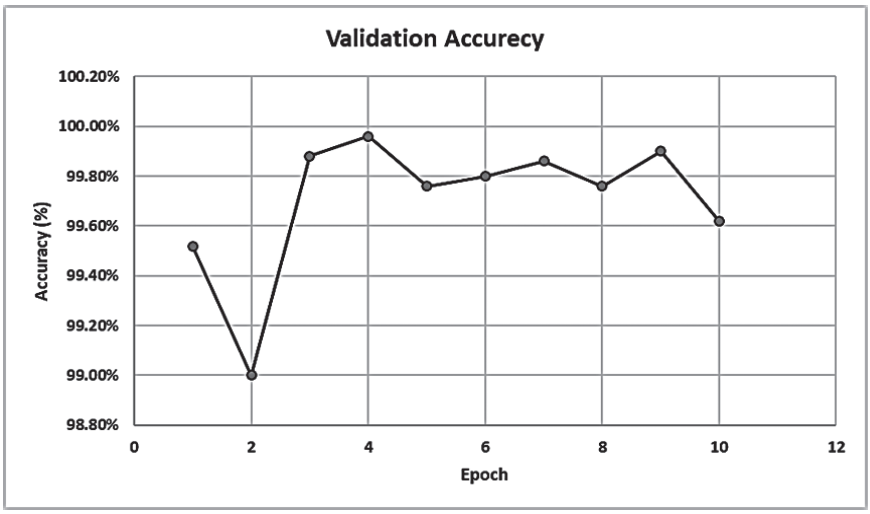

Figure 8
Line chart of the validation accuracy (Word)

When we successfully finished training our CNN model, we predicted some images that were foreign to the data set. Even though our program divided

the data set into training and validation sets, the use of external images bolsters the agility of the CNN architecture. We used a few images that are not present in the data set and showed that our model successfully classified them. To predict these images, we had to preprocess the images according to our CNN requirements.

Table II
Performance of the proposed model

Dataset	Average Validation Loss	Average Accuracy
Alphabet	0.01986	99.40%
Word	0.046877	99.71%

As stated earlier, we have proposed two CNN models to solve a multi-class classification problem. Since Neural Networks try to minimize the error, in both times our training errors have been decreased gradually and in turn accuracy went up.

The results of both models used in our study for American Sign Language (ASL) recognition have shown impressive accuracy rates, exceeding 99%. These models have been trained to predict the meaning of individual ASL signs based on single input images. However, it's worth noting that during our testing, we encountered challenges with correctly classifying external images that were not part of the training data.

We have not yet implemented live classification using cameras in our model due to the need for a larger and more diverse dataset to ensure accurate performance in real-time scenarios. Although we anticipate that the accuracy may decrease to around 80% to 90% when using a larger dataset, we believe our model will still be capable of effectively classifying images captured by a camera.

To improve the robustness of our model, we recognize the importance of incorporating a diverse dataset. This includes images captured within different environments and from various individuals. By incorporating such diversity, we aim to enhance the model's ability to accurately classify ASL signs across a wider range of conditions and contexts.

5. Comparison

We know that in terms of static images, we need to use a convolutional neural network. Whereas for dynamic or time series data, we have to use RNN or LSTM. A huge disadvantage of RNN and updated LSTM is lengthy data preprocessing, as well as prolonged training. So, in our research, we will generate static data from sequential data, which can then be fed into the CNN network. As a result, the time and resources required for preprocessing

will be saved, and training time will also be minimized. We believe that by avoiding the use of RNN architecture, we can achieve similar accuracy to a recurrent neural network while reducing training time. Following table contains the comparison of other works with the proposed in terms of accuracy.

Table III
Comparison of the proposed method to published works

Study	Accuracy [%]
Amer Kadhim and Khamees (2020)	98.53%
Fierro Radilla and Perez Daniel (2020)	96%
Aly, Aly, and Almotairi (2019)	84.5%
Kuznetsova, Leal-Taixe, and Rosenhahn (2013)	87%
Tao, Leu, and Yin (2018)	84.7%
Proposed	**99%**

6. Conclusion

In this research, we have trained two CNNs to recognize the alphabet and some words in the American Sign Language. We minimized the total processing for time series data by utilizing the convolutional neural network. We have evaluated our network for random data from the dataset and showed that by transforming the temporal sequence of data into static data, a convolutional neural network can effectively classify temporal-ordered images. This model provides more than 99% accuracy. We hope that our research will enable non-sign language speakers to communicate with deaf people effectively and easily. In the future, we will include additional words and create a graphic interface for simple understanding.

References

Aly, W., Aly, S., & Almotairi, S. (2019). User-independent American sign language alphabet recognition based on depth image and PCANet features. *IEEE Access*, *7*, 123138-123150. https://doi.org/10.1109/ACCESS.2019.2938829

Atitallah, B. B., Hu, Z., Bouchaala, D., Hussain, M. A., Ismail, A., Derbel, N., & Kanoun, O. (2021). Hand sign recognition system based on EIT imaging and robust CNN classification. *IEEE Sensors Journal*, *22*(2), 1729-1737. https://doi.org/10.1109/JSEN.2021.3130982

Chen, D., Li, G., Sun, Y., Kong, J., Jiang, G., Tang, H., ... Liu, H. (2017). An interactive image segmentation method in hand gesture recognition. *Sensors*, *17*(2), 253. https://doi.org/10.3390/s17020253

Deshmukh, S. G., & Jagade, S. M. (2022). Modelling and optimization of hand gesture recognition using a transfer-learning-based convolutional neural network. *NeuroQuantology*, *20*(22), 1032-1044. https://doi.org/10.48047/nq.2022.20.22.NQ10078

Fierro Radilla, A. N., & Perez Daniel, K. R. (2020). Siamese convolutional neural network for ASL alphabet recognition. *Computación y Sistemas*, *24*(3), 1211-1218. https://doi.org/10.13053/cys-24-3-3481

Kadhim, R. A., & Khamees, M. (2020). A real-time American sign language recognition system using convolutional neural network for real datasets. *TEM Journal*, *9*(3), 937-943. https://doi.org/10.18421/TEM93-14

Ko, S. K., Son, J. G., & Jung, H. (2018, October 9 - 12). *Sign language recognition with recurrent neural network using human keypoint detection*. Paper presented at the RACS '18: International Conference on Research in Adaptive and Convergent Systems, Honolulu Hawaii (pp. 326-328). https://doi.org/10.1145/3264746.3264805

Kuznetsova, A., Leal-Taixé, L., & Rosenhahn, B. (2013). *Real-time sign language recognition using a consumer depth camera*. Paper presenter in the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 83-90).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. https://doi.org/10.1038/nature14539

Liao, Y., Sun, Y., Li, G., Kong, J., Jiang, G., Jiang, D., ... Liu, H. (2017). Simultaneous calibration: A joint optimization approach for multiple kinect and external cameras. *Sensors*, *17*(7), 1491. https://doi.org/10.3390/s17071491

Liu, K., Chen, C., Jafari, R., & Kehtarnavaz, N. (2014). Fusion of inertial and depth sensor data for robust hand gesture recognition. *IEEE Sensors Journal*, *14*(6), 1898-1903. https://doi.org/10.1109/JSEN.2014.2306094

Lu, Z., Chen, X., Li, Q., Zhang, X., & Zhou, P. (2014). A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices. *IEEE Transactions on Human-Machine Systems*, *44*(2), 293-299. https://doi.org/10.1109/THMS.2014.2302794

Luzanin, O., & Plancak, M. (2014). Hand gesture recognition using low-budget data glove and cluster-trained probabilistic neural network. *Assembly Automation*, *34*(1), 94-105.

Masood, S., Srivastava, A., Thuwal, H.C., & Ahmad, M. (2018). Real-time sign language gesture (word) recognition from video sequences using

CNN and RNN. In: Bhateja, V., Coello Coello, C., Satapathy, S., Pattnaik, P. (eds) Intelligent Engineering Informatics. Advances in Intelligent Systems and Computing, vol 695, (pp. 623-632), Springer, Singapore. https://doi.org/10.1007/978-981-10-7566-7_63

O'shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*. https://doi.org/10.48550/arXiv.1511.08458

Saleem, M. M. (2020). Deep learning application on American sign language database for video-based gesture recognition. Published Bachelor degree theses: School of Applied Computing, Faculty of Applied Science and Technology, Sheridan College, Institute of Technology and Advanced Learning, Ontario

Sun, Y., Li, C., Li, G., Jiang, G., Jiang, D., Liu, H., ... Shu, W. (2018). Gesture recognition based on kinect and sEMG signal fusion. *Mobile Networks and Applications*, *23*, 797-805. 97–805. https://doi.org/10.1007/s11036-018-1008-0

Tao, W., Leu, M. C., & Yin, Z. (2018). American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion. *Engineering Applications of Artificial Intelligence*, *76*, 202-213. https://doi.org/10.1016/j.engappai.2018.09.006

Wang, X., Xia, M., Cai, H., Gao, Y., & Cattani, C. (2012). Hidden-Markov-models-based dynamic hand gesture recognition. *Mathematical Problems in Engineering*, *2012*(1), 986134. https://doi.org/10.1155/2012/986134

Wenjin, W., Xiangrong, T., Yun, L., Jingrong, L., Jianyong, C., Xueling, W., ... Hao, W. (2018). Neonatal hearing screening in remote areas of China: A comparison between rural and urban populations. *Journal of International Medical Research*, *46*(2), 637-651. https://doi.org/10.1177/0300060517706643

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, *31*(7), 1235-1270. https://doi.org/10.1162/neco_a_01199

Zhang, Q., & Deng, F. (2017). Dynamic gesture recognition based on leapmotion and HMM-CART model. *Journal of Physics: Conference Series*, *910*(1), 012037. IOP Publishing. https://doi.org/10.1088/1742-6596/910/1/012037