Proceedings of 15th IOE Graduate Conference
Peer Reviewed
Year: 2024 Month: May Volume: 15
ISSN: 2350-8914 (Online), 2350-8906 (Print)

Nepali Sign Language Letter Detection and Finger Spelling Using Mediapipe and CNN

Guptaraj Shrestha [a], Nishan Thing [b], Prajita Dhakal [c], Prasanga Dahal [d], Pukar Karki [e], Manoj Kumar Guragai [f]

[a,b,c,d,e,f] *Department of Electronics and Computer Engineering, Purwanchal Campus, IOE, TU, Nepal*

[a] 076bct034@ioepc.edu.np , [b] 076bct052@ioepc.edu.np , [c] 076bct057@ioepc.edu.np , [d] 076bct058@ioepc.edu.np , [e] pukar@ioepc.edu.np , [f] manojkguragai@ioepc.edu.np

Abstract

This paper describes the development of a machine learning model to translate Nepali Sign Language (NSL) gestures into corresponding Nepali text format. The system utilizes computer vision and deep learning techniques, specifically a Convolutional Neural Network (CNN) model trained on a dataset of over 2,500 images per label with total of 50 labels. The training accuracy was 99%, whereas the validation accuracy was 87.78%. Training loss settled at 1.18%, and validation loss settled at 8.258%. A dropout layer and early stopping function were introduced, and the model was trained for 50 epochs. The model achieved training accuracy of 99.84% and validation accuracy of 99.80%. Similarly, model Training Loss was 0.6% and Validation Loss was 1.16%. The Accuracy curve showed that the both validation and training accuracy gradually increased to 90% for 10 epochs and became steady. Similarly,in Loss curve both validation and training loss gradually decreased to 4% over the course of 10 epochs and stabilized. A Classification table was created using 20% of total dataset with 500 for each label. Model performance was exceptional for most of the labels achieving precision, recall and F1-score close to 1. However,for some labels ,model performance was lower in terms of precision (less than 0.98). The overall accuracy of model was 0.99 describing that model perform well on the entire dataset. The trained model was integrated into a user-friendly web application along with the logic for finger spelling to verify and validate the real life use case of the research.

Keywords

Nepali Sign Language, Machine Learning, Convolutional Neural Network, MediaPipe

1. Introduction

Sign language is an expressive form of communication used by individuals who are deaf or hard of hearing. It allows them to convey thoughts, ideas, and emotions through manual gestures, facial expressions, and body movements. However, deaf individuals often encounter significant barriers in their daily interactions, primarily due to the difficulty of communicating with those who do not understand sign language.

There are numerous sign languages, such as American Sign Language, Indian Sign Language, Nepalese Sign Language, and so on. The signs used in these sign languages are not all the same. In our country, Nepal, various organizations and schools have been assisting deaf people to learn Nepali sign language.

Understanding the importance of connecting the deaf and hearing communities, multiple technologies aimed at improving communication using Nepalese Sign Language for persons who are deaf or hard of hearing were studied. It was found that there weren't any specialized sources online regarding Nepalese Sign Language (NSL). As a result, we aimed to create a model that would translate the NSL into textual output. The user will provide input via webcam, and the model can detect the hand gesture and output the word/alphabet that the user provided as input to the model.

This paper shows a model that can detect 47 Nepali Alphabets, 2 symbols, and one blank. The paper proposes the use of a Convolution Neural Network (CNN). The paper present the use of Google's MediaPipe which is a Framework for building machine learning pipelines. With the help of MediaPipe Hand Model Model, detect hand region and extract keypoints from the hands.

2. Literature Review

Sign language is a crucial communication mode for the deaf and hard-of-hearing community. Automatic sign language recognition (SLR) systems have the potential to bridge the communication gap between these communities and the hearing world. This review focuses on recent advancements in video-based SLR systems that leverage deep learning techniques, particularly Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

Recent literature in sign language recognition highlights a transformative shift towards deep learning methodologies, particularly evident in a pivotal study on Nepali Sign Language [1]. Deep learning is applied for both motion detection and detailed exploration of gestures, extracting spatial and temporal features. Two approaches are explored: one using CNN and RNN, and another using CNN and Vision Transformer. The latter outperforms in accuracy, emphasizing the growing importance of deep learning in enhancing sign language recognition systems. These advancements contribute to improved communication accessibility for individuals relying on sign language.

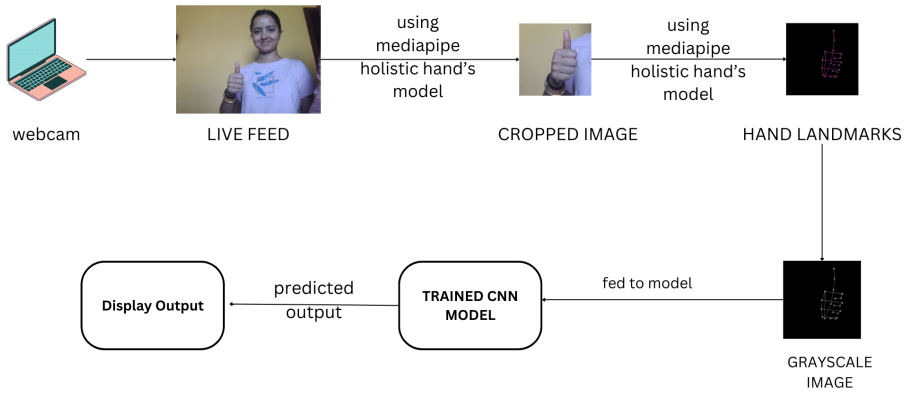

Figure 1: System Block Diagram

Expanding on this foundation, another significant contribution emerged in the form of a novel method for character identification in American Sign Language, utilizing Convolutional Neural Networks (CNNs) [2]. The research not only introduced an innovative approach but also emphasized the indispensable role CNNs play in the realm of sign language recognition, particularly in the context of character identification. The implications of this study extend beyond American Sign Language, shaping the trajectory of gesture recognition in a broader spectrum.

Diversifying the research landscape, a study concentrating on Indian Sign Language harnessed the capabilities of MediaPipe Holistic to identify movements and facilitate their conversion into text or voice, effectively bridging the communication gap between sign language and other modes of expression [3]. Noteworthy is the emphasis on discerning between static and dynamic signs, revealing the superior performance of Long Short-Term Memory (LSTM) models in tracking dynamic phrases, while CNNs demonstrate proficiency in capturing static characters.

In parallel, a groundbreaking project introduced a real-time sign language detection system by synergizing Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [4]. The study, titled "Recognizing Sign Language Gestures from Video Sequences," showcased the synergistic effectiveness of these networks in real-time recognition, setting the stage for practical applications in assistive technology.

Lastly, Research Journal of Engineering and Technology The proposed system integrates Convolutional Neural Networks (CNNs) for visual feature extraction and Long Short-Term Memory (LSTM) networks for understanding the order of signs [5]. The primary objective was to convert Nepali Sign Language into written text. The system captures sign images through a camera and employs a CNN to analyze them. Initial training concentrated on a limited set of signs (5 and 7), with better accuracy achieved in the smaller set. Despite promising outcomes, the study acknowledges constraints, such as a restricted sign vocabulary and reliance on red gloves during testing, posing challenges for practical use. In essence, this research represents a crucial step in utilizing advanced neural networks to enhance communication between the deaf and hearing communities in Nepal.

In summation, the collective body of research signifies the ascendance of deep learning, particularly the integration of CNNs and LSTMs, in the realm of sign language recognition. These studies, encompassing diverse aspects from understanding intricate gestures to real-time detection, depict a vibrant landscape of advancements that are actively shaping the future of communication for the hearing-impaired. The integration of deep learning not only ensures more accurate and responsive gesture recognition systems but also holds the promise of seamlessly incorporating sign languages into mainstream communication technologies.

3. Methodology

3.1 Data Collection

Image datasets were captured using 4 different laptops for 50 labels with an image resolution of 540 x 540 pixels. With the help of the OpenCV library, the image was captured for about 2500 images per label; in total, 1,25,000 image datasets were collected. These captured images were not the final dataset, the aim was to create a dataset containing Hand landmarks for these images. The sample image of the Captured Image is shown in Figure 2.

Figure 2: Original Image

3.1.1 Cropping Capture Image

MediaPipe Hands Model used to crop the image, which is developed by Google to utilizes an ML pipeline consisting of

multiple models working together: A palm detection model that operates on the full image and returns an oriented hand bounding box and a hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints[6] as shown in Figure 3.

The image with resolution 540 x 540 px was passed through the MediaPipe hand model with parameter static_image mode = True, min_detection_confidence=0.3, and max_num_hands=1 to detect the palm area from the captured image. Before being fed to the Hand landmark model, the image is converted into RBG format. Using this model, a set of 21 key points representing various landmarks on the hand was obtained, such as fingertips, knuckles, and the palm. Based on the Hand Landmark index 9 i.e MIDDLE FINGER MCP and Hand Landmark index 12 i.e. MIDDLE FINGER TIP, the original image was cropped. The size of the cropped image was set to 200 x 200 pixels, determined by the original dimensions of the image. These cropped images contain only the image of one palm or hand, which is in BGR form. The sample of the cropped image is shown in figure 4. This cropped image contained only the palm area image not a Hand Keypoints.

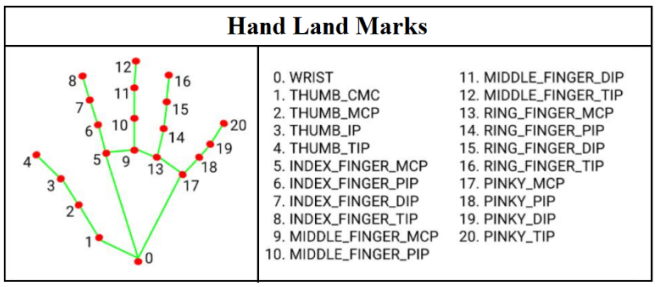

Figure 3: Hand Landmark

3.1.2 Hand Landmark Image Drawing

The cropped image was converted into RGB form and fed to MediaPipe Hand model to extract key points of hand from the cropped images. Based on the coordinates of the extracted key points, the coordinates of landmarks were drawn on the plain black background. The drawn hand landmark image had the same shape as the cropped images i.e. 200 x 200 pixels. The landmark image was saved in grayscale format. The sample image of the Landmark image is shown in figure 5. This Hand Landmark Image was the final Dataset. Each Character of the Nepali Alphabet was labeled as shown in figure 6.

Figure 4: Cropped Image

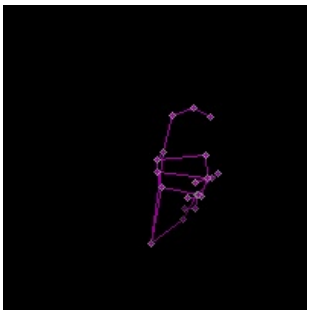

Figure 5: Landmark Image

Letter	अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	क	ख	ग	घ	ङ
Label	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Letter	च	छ	ज	झ	ञ	ट	ठ	ड	ढ	ण	त	थ	द	ध	न
Label	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Letter	प	फ	ब	भ	म	य	र	ल	व	श	ष	स	ह	क्ष	त्र
Label	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
Letter	ज्ञ	श्र	़	़	blank										
Label	45	46	47	48	49										

Figure 6: Dataset Label

3.2 Data Pre-Processing

During pre-processing, the dataset image was converted into a NumPy array and reshaped to 200 x 200 pixels. The pixel values lie within a range of 0 to 1 in this preprocessed image data, which comprises a 4D NumPy array.

3.3 Training CNN Model

The training process focused on instructing the model to identify Nepali Sign Language (NSL) signs from video frames of gestures. Initially, 125,000 total dataset were divided into training and validation sets with an 80-20 ratio using the split folders library i.e. Training set consists of 100,000 dataset and the Validation or Test set consists of 25,000 dataset. ImageDataGenerator was employed with an input size of 200*200, class mode set to 'categorical', color mode set to 'grayscale', and a batch size of 128. This was utilized to convert the data into the necessary format and normalize the dataset using the rescale parameter.

In order to include all required layers, the model's architecture contained a sequential Keras model. At first, features were extracted from the video frame using Conv2D layers with a 3x3 kernel and the 'relu' activation function. The MaxPooling2D layers with a 2x2 filter were applied before each Conv2D layer, allowing for a reduction in both image size and complexity. To reduce overfitting, dropout layers with a 0.4 dropout rate were also included. To improve the model's compatibility, these three layers were replicated. The 4D data was then transformed into 1D using a flatten layer. 'Relu' activation functions were used in two Dense layers, with 256 and 64 channels, respectively. The final classification of NSL signs was given to a Dense layer with 50 output channels and softmax activation.

To compile the model, the Adam optimizer with a default learning rate of 0.001 was employed. Other optimizers, such as SGD with a learning rate of 0.01 and RMSProp with a learning rate of 0.001, were also tested, but Adam was found to be the best fit for the model. To find the optimal parameter, the learning rate of Adam was further revised, with values ranging from 0.0001, 0.002, and 0.0002, but the default Adam setting turned out to be the most accurate. For the loss function, Categorical Cross-Entropy was utilized. Additionally, accuracy was used as a statistic to evaluate how well the model predicted NSL signals. The model was then trained using the fit function, which was used to select the number of epochs and incorporate the validation set to measure performance throughout the process. The validation loss was

analyzed using the EarlyStopping function, which was used to end the fitting process when suitable. Finally, the accuracy of the model was evaluated to measure its effectiveness.

3.4 Model Integration with Logic for Finger Spelling

A straightforward approach was developed to combine letters detected by a CNN model for Nepali finger spelling. Letters are stored in an array, and once the array is full and contains at least two letters, they are processed to form a complete word. The system differentiates between consonants and vowels, applying specific logic based on their combinations. Vowel-vowel pairs are simply joined, while consonant-vowel combinations involve replacing the vowel with its corresponding symbol as shown in figure 7 and joining it with the consonant. In both vowel-consonant and consonant-consonant combinations, the initial part (or the first letter in the array) remains unchanged. The process then focuses on modifying the second consonant based on the following letter (third letter in the array for any of the above combinations). This logic ensures proper Nepali word formation by considering the unique characteristics of consonant-vowel interactions in the language.

Vowel	अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ
Symbol	अ	ा	ि	ी	ु	ू	े	ै	ो	ौ

Figure 7: Vowel and Their Symbol

4. Results and Discussion

4.1 Result

4.1.1 Accuracy and Loss Curve

The accuracy curve shown in figure 8 rapidly increases to nearly 100%, indicating how well the model fits the training set of data. However, the validation accuracy curve peaks around 90% after an initial sharp rise. This indicates that the model does not get any more accurate with more training epochs, but it does generalize to unknown data very well initially.

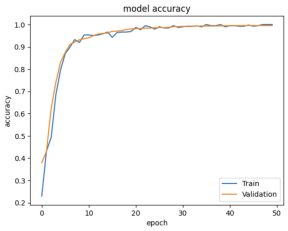

Figure 8: Accuracy Curve

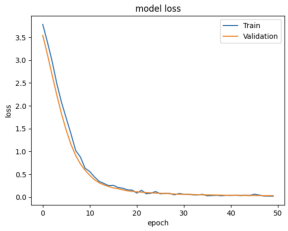

Figure 9: Loss Curve

Similarly, the Loss Curve shown in figure 9 drops sharply to near zero within the first few epochs. This confirms the model fits the training data extremely well after initial learning. However, the validation loss curve initially decreases and then peaks at a higher value compared to the training loss. This indicates the model is being overfitted to the training data. This limits the model's ability to generalize and perform well on new, unseen data from the validation set.

4.1.2 Classification Table

	precision	recall	f1-score	support
0	0.98	1.00	0.99	500
1	0.97	0.99	0.98	500
2	0.99	0.93	0.96	500
3	0.99	1.00	0.99	500
4	1.00	0.99	0.99	500
5	1.00	1.00	1.00	500
6	1.00	1.00	1.00	500
7	1.00	1.00	1.00	500
8	1.00	0.99	0.99	500
9	0.99	0.99	0.99	500
10	0.98	1.00	0.99	500
11	0.99	1.00	1.00	500
12	1.00	1.00	1.00	500
13	0.99	0.98	0.98	500
14	0.98	0.98	0.98	500
15	0.99	0.98	0.99	500
16	0.99	1.00	1.00	500
17	0.98	1.00	0.99	500
18	1.00	0.99	0.99	500
19	0.96	0.99	0.97	500
20	1.00	1.00	1.00	500
21	0.97	1.00	0.99	500
22	0.99	1.00	0.99	500
23	1.00	1.00	1.00	500
24	0.99	0.96	0.98	500
25	1.00	0.99	0.99	500
26	1.00	1.00	1.00	500
27	0.99	0.99	0.99	500
28	1.00	1.00	1.00	500
29	0.97	0.99	0.98	500
30	0.99	0.99	0.99	500
31	1.00	1.00	1.00	500
32	1.00	0.99	0.99	500
33	1.00	1.00	1.00	500
34	1.00	1.00	1.00	500
35	1.00	0.97	0.98	500
36	0.99	1.00	0.99	500
37	1.00	1.00	1.00	500
38	0.97	1.00	0.99	500
39	0.98	1.00	0.99	500
40	1.00	1.00	1.00	500
41	1.00	1.00	1.00	500
42	1.00	1.00	1.00	500
43	1.00	0.99	1.00	500
44	1.00	1.00	1.00	500
45	1.00	0.97	0.98	500
46	0.99	0.98	0.99	500
47	1.00	1.00	1.00	500
48	0.98	0.97	0.97	500
49	0.98	0.98	0.98	500
accuracy			0.99	25000
macro avg	0.99	0.99	0.99	25000
weighted avg	0.99	0.99	0.99	25000

A classification report is a comprehensive summary of the performance of a classification algorithm on a dataset. It provides various metrics for each class, helping to evaluate the model's precision, recall, F1 score, and support. The classification Table of our model is shown in the above Table.

4.1.3 Output

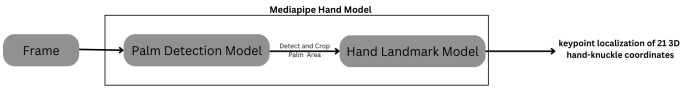

Figure 10: Hand Landmark Detection Steps

Figure 11: Output

Figure 11 shows the real time prediction of a letter and joining of each letter to form a word at last.

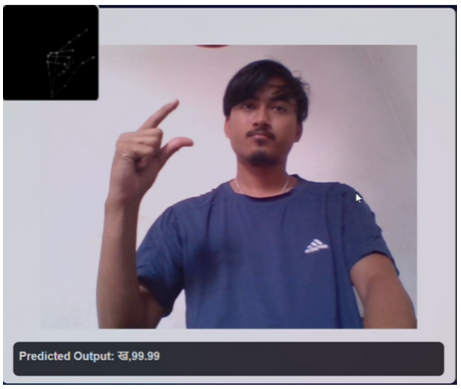

Figure 12: Success Case

Hand gesture is detected and landmark is successfully drawn, but only when the hand is shown to the camera at the right angle, at about 60 cm from the camera, the hand must be held stable, and the signs must only be given according to prompts displayed on the screen.

Figure 13: Change in Angle

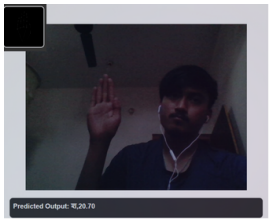

Figure 14: Low Light

Hand might not be detected properly if it's too close to the camera or if it's not facing directly towards the camera. Changes in lighting can also affect detection. In case of wrong hand gesture input, it might give the wrong output.

5. Conclusion

The majority of the deaf community faces deprivation of basic services due to communication barriers with the hearing community. This paper introduced a model designed to predict Nepali Sign language characters. Mediapipe and OpenCV were used for Real-time data collection. Model-building strategies like transfer learning and different neural network architectures were examined and a simple CNN model which contained ConV2D, MaxPolling layer, Flatten layer, and Dense layer, was ultimately chosen. Optimizers like 'Adam', 'RMSProp', and 'SGD' were examined, leading to the selection of Adam optimizers. A graph was analyzed by changing the learning rate to 0.0001, 0.002, 0.0002, etc. However, the default 'Adam' worked accurately for the model. At first, Relu activation function was used in all layers including the output Dense layer. The Accuracy and Loss curve were steady. Finally Softmax activation function was used which provided the appropriate graph with accuray of 99%.The model's advancement to produce words directly instead of only characters is a crucial step toward its future development. The model needs to be improved for real-time processing speed in order for its use to be expanded beyond primary school students. Furthermore, adding extra features like captioning for sign language videos can significantly improve inclusion and accessibility. All of these improvements work together to improve the model's performance and increase the range of educational levels and communication modalities in which it can be used.

Acknowledgments

The authors extend their heartfelt gratitude to the Department of Electronics and Computer Engineering, Purwanchal Campus for the unwavering support, guidance, and encouragement.

References

[1] S. Ligal and D. S. Baral. Nepali sign language gesture recognition using deep learning. In *Proceedings of the 12th IOE Graduate Conference*, volume 12, October 2022.

[2] Sarfaraz Masood, Harish Chandra Thuwal, and Adhyan Srivastava. American sign language character recognition using convolution neural network. In *Smart Computing and Informatics*, pages 403–412. Springer, 2018.

[3] Kaushal Goyal and Dr. Velmathi G. Indian sign language recognition using mediapipe holistic. In *Proceedings of the VIT*, Chennai, India, 2023.

[4] Sarfaraz Masood, Adhyan Srivastava, Harish Chandra Thuwal, and Musheer Ahmad. Real-time sign language gesture (word) recognition from video sequences using cnn and rnn. In *Intelligent Engineering Informatics*, pages 623–632. Springer, 2018.

[5] D. Mali, R. Mali, S. Sipai, and S. P. Pandey. Nepali sign language translation using convolutional neural network. In *1st KEC Conference Proceedings, Volume I*, September 27 2018.

[6] MediaPipe. Mediapipe hands documentation, 2024.