Journal of Innovative Image Processing (JIIP) (2020)
Vol.02/ No. 02
Pages: 65-76
https://www.irojournals.com/iroiip/
DOI: https://doi.org/10.36548/jiip.2020.2.001

Real Time Sign Language Recognition and Speech Generation

Amrita Thakur, Pujan Budhathoki, Sarmila Upreti, Shirish Shrestha, Subarna Shakya*
Department of Electronics and Computer Engineering, Pulchowk Campus, Nepal.
*Email: drss@ioe.edu.np

Abstract: Sign Language is the method of communication of deaf and dumb people all over the world. However, it has always been a difficulty in communication between a verbal impaired person and a normal person. Sign Language Recognition is a breakthrough for helping deaf-mute people to communicate with others. The commercialization of an economical and accurate recognition system is today's concern of researchers all over the world. Thus, sign language recognition systems based on Image processing and neural networks are preferred over gadget system as they are more accurate and easier to make. The aim of this paper is to build a user friendly and accurate sign language recognition system trained by neural network thereby generating text and speech of the input gesture. This paper also presents text to sign language generation model that enables a way to establish a two-way communication without the need of a translator.

Keywords: American Sign Language (ASL), Sign Language Recognition; Convolutional Neural Network (CNN); Python; VGG-16

1. Introduction:

Sign Language is a gesture-based language which involves hand movements, hand orientation and facial expression instead of acoustic sound patterns. This type of language has varying patterns according to the people and is not universal. Nepali sign language differs from Indian, American and also with in different parts of Nepal. However, since most people don't have prior knowledge of sign language of any sort, it becomes harder and harder for deaf-mute people to communicate without a translator, thus they feel ostracized. Sign Language Recognition has been accepted as a widely recognized communication model between deaf-mute people and normal people. Recognition models are categorized under computer vision-based and sensor-based systems. In computer vision-based gesture recognition, camera is used for input and image processing of input gestures is done before recognition. The processed gestures then are recognized using various algorithms like Hidden Markov Model and Neural network techniques. The main drawback of vision-based sign language recognition system image acquisition process has many environmental apprehensions such as the place of the camera, background condition and lightning sensitivity. But it is easier and more economical than using sensor and tracker for data. However, Neural Network techniques and Hidden Markov Model are used together with sensor data for more accuracy.

This paper works on computer vision-based hand gesture recognition using convolutional neural network on Python. The major dataset up-to now is American Sign Language alphabets. The datasets of gesture are preprocessed using Python libraries and packages like OpenCV and skimage, then trained using CNN VGG-16 model. The recognized input is converted into speech. This will provide one-way communication as a person who doesn't understand sign language will get the meaning of the hand signs shown to him/her. Furthermore, to make two-way communication possible, this paper also

Journal of Innovative Image Processing (JIIP) (2020)
Vol.02/ No. 02
Pages: 65-76
https://www.irojournals.com/iroiip/
DOI: https://doi.org/10.36548/jiip.2020.2.001

presents text to sign language conversion, which allows a person, who doesn't understand sign language, to convert text to sign language finger spelling that the signer would understand.

2. Related Works:

Many vision-based and sensor-based techniques have been used for sign language recognition. Pavlovic *et al*. [1] discussed about the visual interpretation of hand gestures for Human-Computer Interaction. The paper published on 1997 emphasizes on the advantages and shortcomings and important differences in the gesture interpretation approaches depending on whether a 3D model of the human hand or an image appearance model of the human hand is used. As of the time, this survey was done 3D hand models offered a way of more elaborate modeling of hand gestures but lead to computational hurdles that had not been overcome given the real-time requirements of HCI. They also discussed implemented gestural systems as well as other potential applications of vision-based gesture recognition.

Jiyong *et al.* [2] presented as input to the Hidden Markov Models (HMMs) for a real-time system designed to recognize continuous Chinese Sign Language (CSL). Raw Data was collected from two Cyber-Gloves and a 3-D tracker. Dynamic programming (DP) technique was used to segment the training sentence into basic units, then; estimating was done by the Welch-Baum algorithm. Test results using 220 words and 80 sentences, and the system showed 94.7% recognition rates.

Volger *et al.* [3] used Parallel Hidden Markov models (PaHMMs) for American Sign Language recognition. They stated that phonemes could be used instead of whole signs for a continuous recognition system. Used Two channels to the right and left hands, assuming any word can be broken down into fundamental phonemes the same as words in speech recognition. A single channel of the HMM model was tested for a small vocabulary number (22 signs) with results showing an 87.88% accuracy rate. The system cannot recognize a larger vocabulary, hand configuration, orientation, and facial expressions.

Gunasekaran *et al.* [4] in their paper discussed about the recognition of sign language and translation into speech using a microcontroller system. The proposed system played a pre-recorded voice every time a sign language gesture was detected. The proposed model consisted of four modules, they are sensing unit, processing unit, voice storage unit, and a wireless communication unit. It was achieved by integrating the flux sensor and APR9600 with PIC16F877A. The flux sensors are placed in gloves, which respond to the gesture. By using a suitable circuit response of the sensor was given to the microcontroller based on the response microcontroller played the recorded voice using APR9600. This system offered high reliability and fast response.

The paper by Kalidolda *et al.* [5] described the project, which aimed to develop an interpreting robotic system of sign language. The project had two core objectives of facilitating fingerspelling recognition in real-time and conducting performance testing „in the wild" and to get feedback on the NAO robotic interpreting system from deaf-mute individuals. The robotic system comprises a number of software

Journal of Innovative Image Processing (JIIP) (2020)
Vol.02/ No. 02
Pages: 65-76
https://www.irojournals.com/iroiip/
DOI: https://doi.org/10.36548/jiip.2020.2.001

and hardware components, particularly: Microsoft Kinect SDK for human detection and tracking, Leap Motion SDK for hands detection and tracking. A humanoid stationary NAO robot (NaoQi SDK) acting as a social interface. NAO also takes the role of a robotic tutor for children, A humanoid stationary Pepper robot (Pepper SDK for An-droid) acting as a social interface, A computer monitor for virtual robot and Android tablet (Android SDK) is used for tutoring applications.

3. Proposed Work

The recognition of sign language gestures from real time video and successfully classifying it into either one from a list of categories have been a popular and challenging field of research. Many researchers have been working on this field for a long time, so we have also thought of contributing to this field as by working on it in our final year major project. Liang *et al.* [6] have also put their research on this concept which has guided us throughout the implementation. The process of recognizing a sign language gesture and classifying it is the one line definition of the task performed by this proposed system. Along with this, a *text to ASL finger spelling* feature is also available that makes the two-way communication from *sign to text and text to sign* possible. The following steps were taken while working on this project:

3.1 Data Acquisition

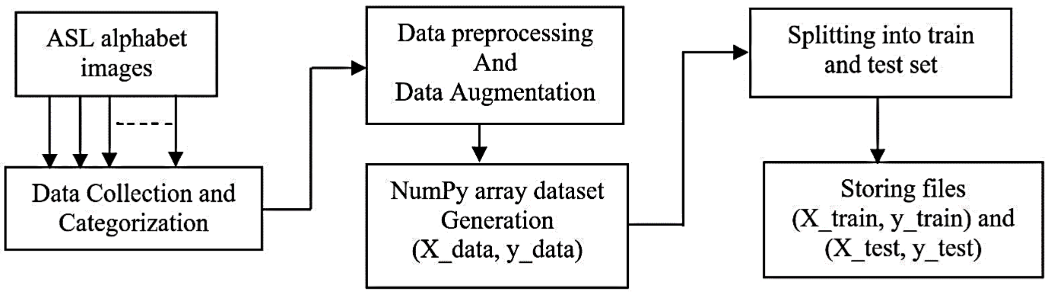

ASL alphabet images
Data Collection and Categorization
Data preprocessing And Data Augmentation
NumPy array dataset Generation (X_data, y_data)
Splitting into train and test set
Storing files (X_train, y_train) and (X_test, y_test)

Figure 1: Block diagram of data acquisition and train-test dataset generation

The images of various alphabets of American Sign Language were collected using different webcams from different laptops. At first, *a data collection program* was created using OpenCV and Pillow library packages in Python. This program has two versions: first one being the *manual image capture version*, in which all the images were captured manually with varying background and hand poses. Since it was time consuming, we ended up making a second version,*video capture and split frames version*, which would capture the video of the hand gesture in different background and automatically split it into the desired number of frames. The dataset is created by placing respective images for various alphabets inside a folder named after that alphabet, for instance, all the images collected for a category "S" are placed inside a folder named "S", the folder name acts as the labels for training the dataset so this is important. With these things in mind, the current dataset is being created. And finally, captured images are also resized to 202 by 202 pixels.

Journal of Innovative Image Processing (JIIP) (2020)
Vol.02/ No. 02
Pages: 65-76
https://www.irojournals.com/iroiip/
DOI: https://doi.org/10.36548/jiip.2020.2.001

Then, after that, we started working on *a data preprocessing program* that would convert the resized images to grayscale images. We collected in total 2500 images for each alphabet till now. Then, we shuffled the collected dataset for each category to generate a random dataset for each category and used *data augmentation processes* to get another 2500 images by horizontal flipping 1000 images, adding Gaussian noise to 600 images and adding Gamma contrast to 900 images using imageio and imgaug libraries in Python. After this, the datasets for various categories were converted into NumPy array data of shape (50, 50, 3) for training the dataset using CNN model. At first, we created X_data and y_data that represented all the images in the dataset and their respective labels. Then, we did a train-test split, and obtained (X_train,y_train) and (X_test, test) as our training and testing NumPy array files. Currently the dataset consists of alphabets from A to H and another alphabet dataset are still being completed. We also generated a *category list* that had all the category names i.e. folder names such as 'A','B'… 'H'. This was used to create a dictionary that would map numbers starting from 0 to (number of categories -1) to the alphabets from 'A' to other categories. That is, 0 is mapped to 'A', 25 is mapped to 'Z' and so on if other categories are added.

3.2 CNN model generation and training:

After generating a train and test dataset along with labels for both, all in NumPy array format, we make a Convolutional Neural Network, so that we can train it with the ASL alphabet dataset. Convolutional Neural Network is a class of deep, feed-forward artificial neural networks, most commonly applied to analyze visual imagery. CNNs use a variation of multilayer perceptron designed to require minimal preprocessing. Convolutional Neural Network Model was proposed by LeCun *et al.* [7], and has proved to be a significant milestone in the area of detection and classification of imagesWe use the VGG-16 pretrained model as the foundation for making our model. The model we have is slightly different than a pretrained VGG-16 model. We removed its top input layer so that we can give it an input image with shape (50,50,3) and another change we made is by making the output layer classify the images into the obtained number of classes. As we can see in the figure above, the inputimage of shape (50,50,3) is passed as a NumPy array to the model. Then, it is passed through two or more convolutional layers, where the convolution operation takes place and after that max pooling operation is performed. This combination is repeated for five times. After that, flattening operation is performed and at the end classification is done on the basis of the list of categories present.

We use the saved train data file, X_train.npy, and labels file, y_train.npy, to train the model in Google Colaboratory. At first, we use Google Drive to load all the necessary files including X_train, X_test, y_train, y_test and the pretrained VGG-16 model. After that, we encode the label data i.e. y_train and y_test data using one hot encoding that is label 2 would be represented as [0,0,1,0,0,…,0,0], the length

ISSN: 2582- 4252 (online)

.

of this list would be equal to the number of categories. Then, we use the model that was described in 3.2. to train on the ASL alphabet dataset using Keras TensorFlow and scikit learn libraries in Python.

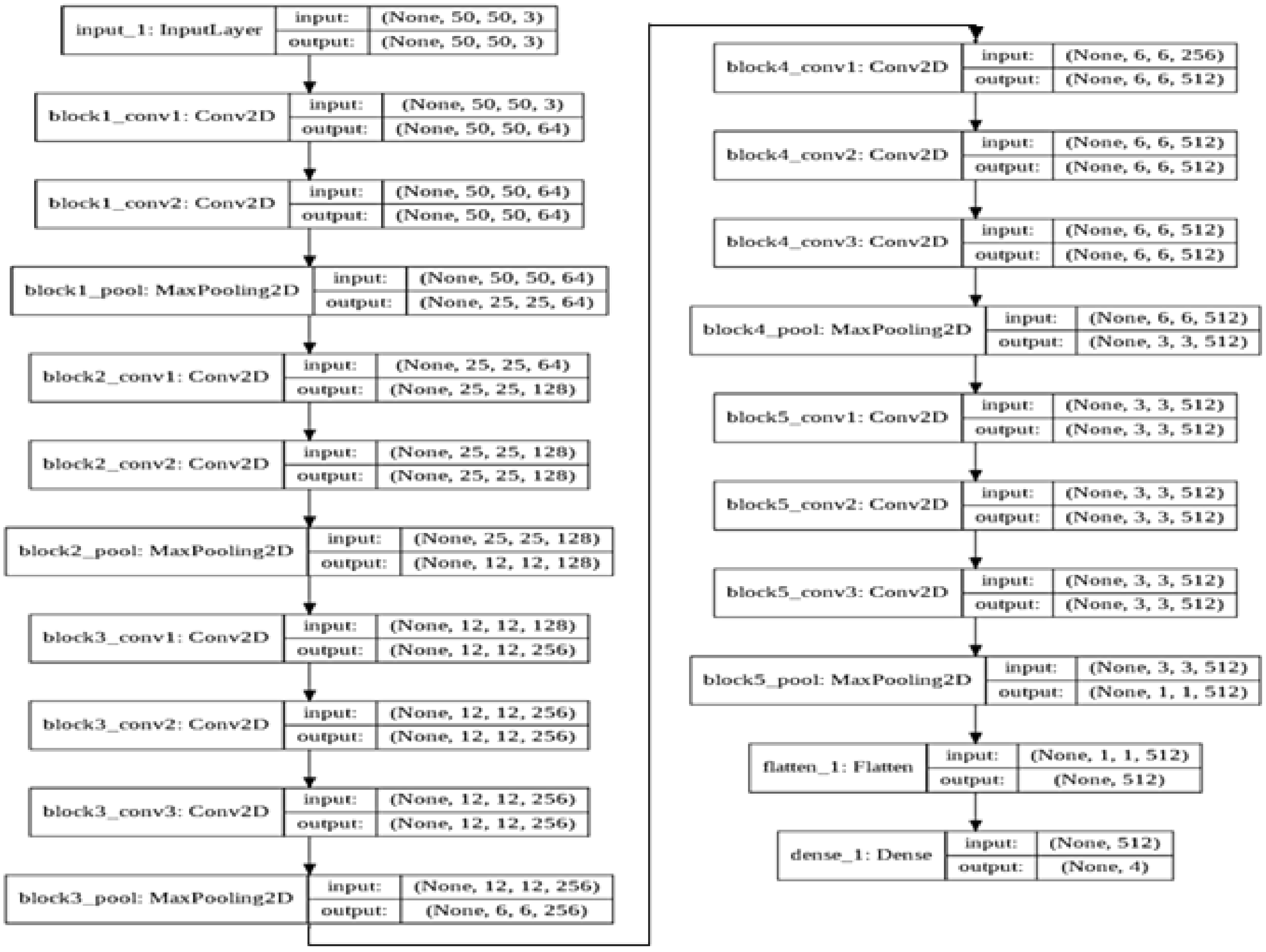

input_1: InputLayer | input: (None, 50, 50, 3) | output: (None, 50, 50, 3)
block1_conv1: Conv2D | input: (None, 50, 50, 3) | output: (None, 50, 50, 64)
block1_conv2: Conv2D | input: (None, 50, 50, 64) | output: (None, 50, 50, 64)
block1_pool: MaxPooling2D | input: (None, 50, 50, 64) | output: (None, 25, 25, 64)
block2_conv1: Conv2D | input: (None, 25, 25, 64) | output: (None, 25, 25, 128)
block2_conv2: Conv2D | input: (None, 25, 25, 128) | output: (None, 25, 25, 128)
block2_pool: MaxPooling2D | input: (None, 25, 25, 128) | output: (None, 12, 12, 128)
block3_conv1: Conv2D | input: (None, 12, 12, 128) | output: (None, 12, 12, 256)
block3_conv2: Conv2D | input: (None, 12, 12, 256) | output: (None, 12, 12, 256)
block3_conv3: Conv2D | input: (None, 12, 12, 256) | output: (None, 12, 12, 256)
block3_pool: MaxPooling2D | input: (None, 12, 12, 256) | output: (None, 6, 6, 256)
block4_conv1: Conv2D | input: (None, 6, 6, 256) | output: (None, 6, 6, 512)
block4_conv2: Conv2D | input: (None, 6, 6, 512) | output: (None, 6, 6, 512)
block4_conv3: Conv2D | input: (None, 6, 6, 512) | output: (None, 6, 6, 512)
block4_pool: MaxPooling2D | input: (None, 6, 6, 512) | output: (None, 3, 3, 512)
block5_conv1: Conv2D | input: (None, 3, 3, 512) | output: (None, 3, 3, 512)
block5_conv2: Conv2D | input: (None, 3, 3, 512) | output: (None, 3, 3, 512)
block5_conv3: Conv2D | input: (None, 3, 3, 512) | output: (None, 3, 3, 512)
block5_pool: MaxPooling2D | input: (None, 3, 3, 512) | output: (None, 1, 1, 512)
flatten_1: Flatten | input: (None, 1, 1, 512) | output: (None, 512)
dense_1: Dense | input: (None, 512) | output: (None, 4)

Figure 2: CNN model used for training on ASL alphabet dataset

3.3 Testing the classification using the model:

For testing whether the system was rightly classifying the image, provided to it at real time, we used two methods. The first method uses the *manual image capture* as done in the data collection program,here, the signer would place his/her hand in a blue 202x202 box displayed on the screen, and press "space" to capture an image, then that image would be converted to a NumPy array and using the methods in Keras library for predicting the classification, we finally obtain *a category index* like 0,1,2...(number of categories – 1), this index would be used as a key in dictionary that would map the index value to category. Like, 1 would map to B and 7 would map to H

Journal of Innovative Image Processing (JIIP) (2020)
Vol.02/ No. 02
Pages: 65-76
https://www.irojournals.com/iroiip/
DOI: https://doi.org/10.36548/jiip.2020.2.001

Although this method worked well most of the times, it still got the classification wrong many times, so we decided on also using a second method so that the classification would be more accurate. So, the second method we used is like *video capture and split frames* method in data collection program, but here we capture a video of the hand sign enclosed by the blue box, and then split it into 100 images, it is better to keep the hand sign within the box and keep its position changing, so that images with different position for the same hand sign can be captured and a list of classification would be generated. The most frequent classification would be taken, that would be the alphabet output for the hand sign. This method requires more processing time than the previous method and the reasons are quite obvious,

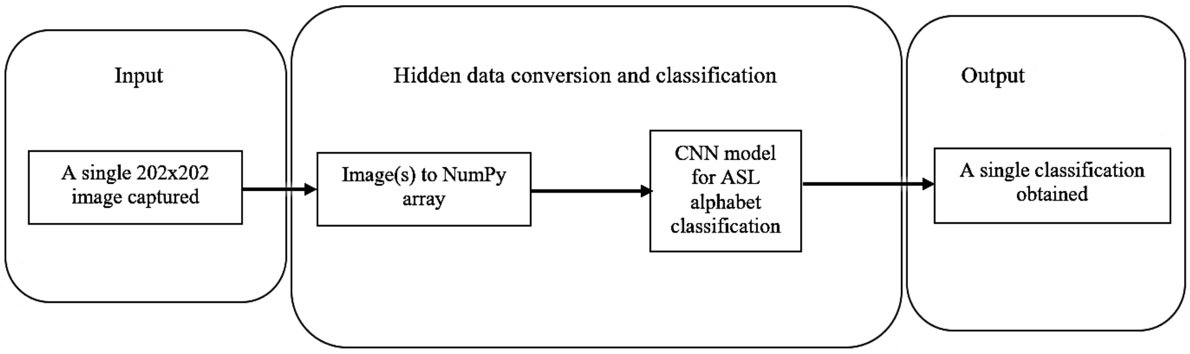

Input
A single 202x202 image captured
Hidden data conversion and classification
Image(s) to NumPy array
CNN model for ASL alphabet classification
Output
A single classification obtained

Figure 3: Manual Image capture and classify method

still this method shows wrong classification but a smaller number of times than before.

For these methods to work, the image should be captured in a well-lit room and the image background should be plain if possible. Furthermore, it is good to place the hand sign inside the blue box such that most area is covered by the hand instead of the background. Thus, this concludes the proposed work for the

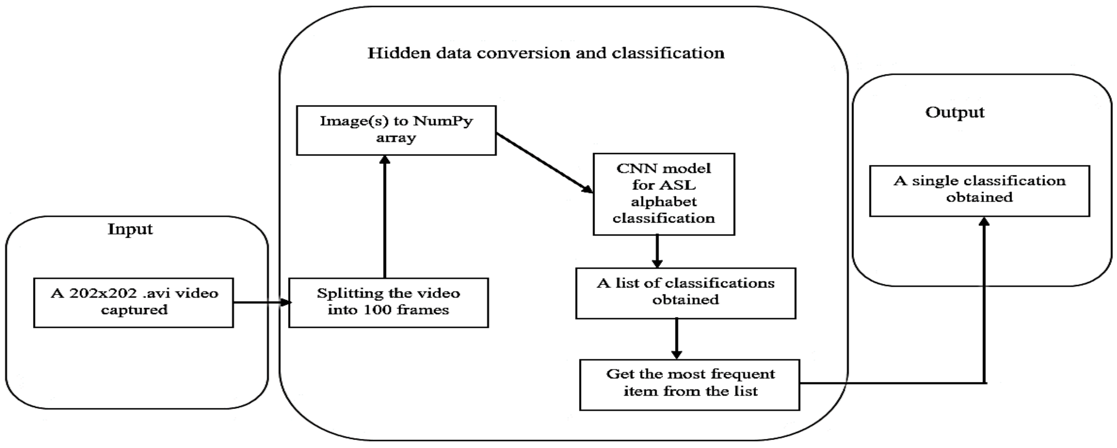

Hidden data conversion and classification
Image(s) to NumPy array
CNN model for ASL alphabet classification
Output
A single classification obtained
Input
A 202x202 .avi video captured
Splitting the video into 100 frames
A list of classifications obtained
Get the most frequent item from the list

Figure 4: Video capture-split frame and classify method

system that generates text, by classifying the input image using the described CNN model, which can be converted to speech using Python library package pyttsx.

Journal of Innovative Image Processing (JIIP) (2020)
Vol.02/ No. 02
Pages: 65-76
https://www.irojournals.com/iroiip/
DOI: https://doi.org/10.36548/jiip.2020.2.001

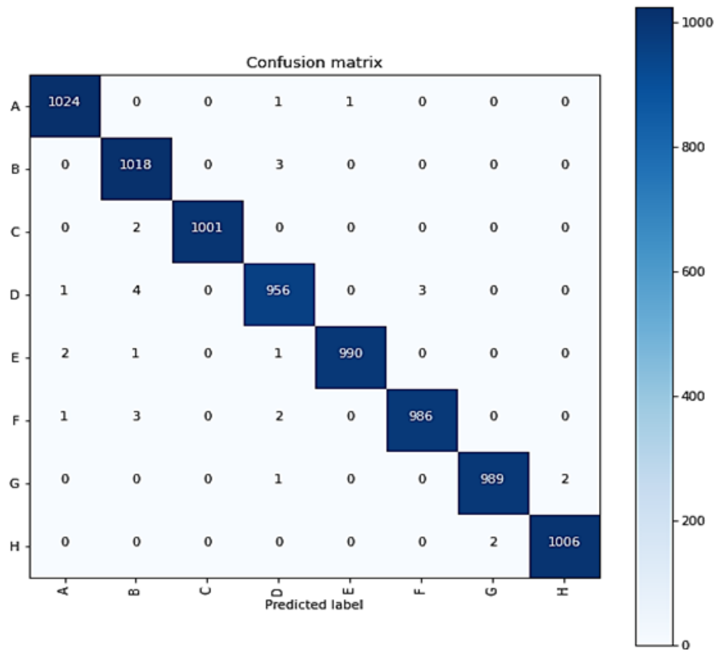
3.4 Text to ASL finger spelling:

This is an additional feature to the ASL recognition system and allows a two-way communication between the people who know sign language and people who don't. This is a rather simple system and has a lot of space for improvements. This system takes a *string with alphabets and spaces* like "Hello", "Sherlock Holmes", etc. as an input. There is a local directory that has images for ASL alphabets from A to Z, although J and Z are both motion-based signs, we have used a static frame from the motion-based representation that depicts the alphabet the most. The images are named after the alphabets that A.jpg to Z.jpg. The obtained input string is converted to a list that has the strings that were separated by a space as its elements. That is, *"Sherlock Holmes" becomes ["Sherlock","Holmes"]*. Then, we do string manipulation and image retrieval by going through each string element and getting the corresponding image, forming a new list with Image data type using Pillow Python package. And using this list to store the result as a GIF format and also display it using OpenCV.

4. Results and Discussion

The model proposed by Simonyan *et.al* [8], VGG-16 is a convolutional neural network architecture having 13 convolutional layers, 5 max pooling layers and 3 dense layers which sums up to 21 layers but has only 16 weight layers, thus its name. It was used to win ILSVR (ImageNet) competition in 2014. Conv 1 has 64 filters as 64 while Conv 2 has 128 filters, Conv 3 has 256 filters while Conv 4 and Conv 5 has 512 filters. VGG-16 network is trained on ImageNet dataset which has over 14 million images and 1000 classes, and achieves 92.7% top-5 accuracy. It surpasses AlexNet network by replacing large filters of size 11 and 5 in the first and second convolution layers with small size 3x3 filters.

In this project, we used the Google Colaboratory TPU to train the model with the ASL alphabet data. The loss function used is Categorical Cross entropy and the optimizer used is Adam optimizer with default learning rate. The model is training for 20 epochs on the training dataset with batch size as 32 and using test dataset as the validation dataset. The dataset consists of images of alphabets from A to H. The training accuracy and training loss were obtained as 99.65 % and 0.0259 respectively. And the test accuracy was 99.62%. We also plotted the training and test accuracy curve along with training and test loss curve. And, we also obtained a confusion matrix and calculated precision, recall, F1-score and support using scikit learn library in Python. The total number of images used to train the model was 32,000 and the test dataset had 8,000 images.



```
              precision    recall  f1-score   support

           A       1.00      1.00      1.00      1026
           B       0.99      1.00      0.99      1021
           C       1.00      1.00      1.00      1003
           D       0.99      0.99      0.99       964
           E       1.00      1.00      1.00       994
           F       1.00      0.99      1.00       992
           G       1.00      1.00      1.00       992
           H       1.00      1.00      1.00      1008

    accuracy                           1.00      8000
   macro avg       1.00      1.00      1.00      8000
weighted avg       1.00      1.00      1.00      8000

=============================================================
Y_pred_classes: [6 3 5 ... 3 0 7]
=============================================================
Y_true [6 3 5 ... 3 0 7]
=============================================================
```

Figure 6: Precision, recall, F1-score and support for various alphabets

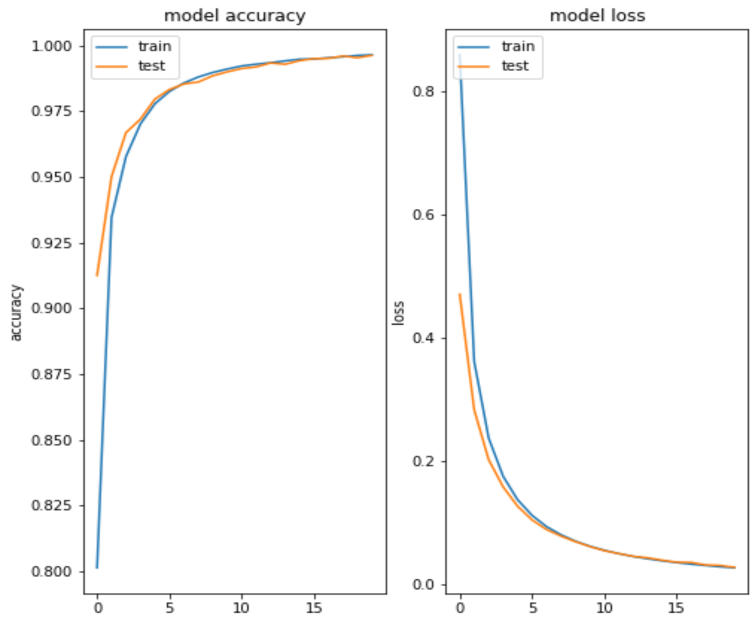
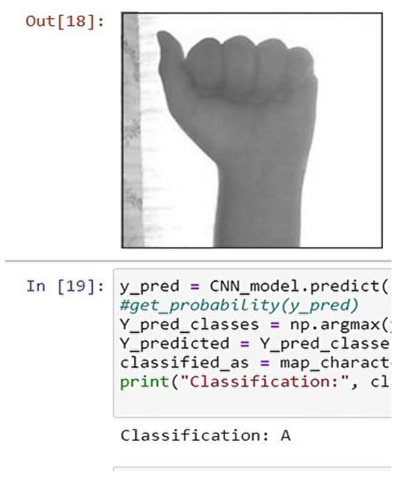
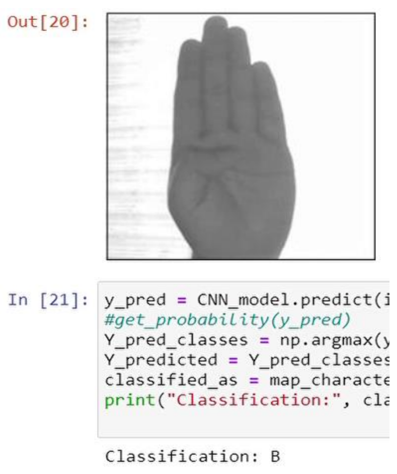


Figure 7: Training and testing data accuracy and loss curves

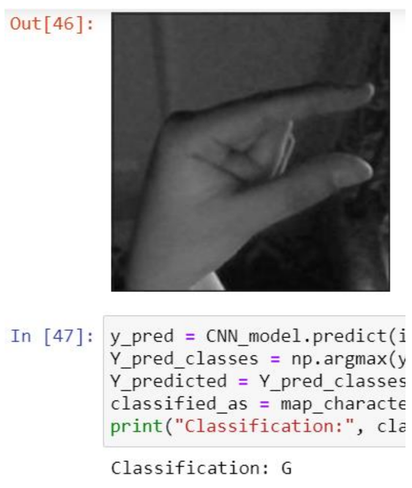
4.1 Results for ASL recognition system:



```
Out[18]:

In [19]: y_pred = CNN_model.predict(
         #get_probability(y_pred)
         Y_pred_classes = np.argmax(
         Y_predicted = Y_pred_classe
         classified_as = map_charact
         print("Classification:", cl

         Classification: A
```



```
Out[20]:

In [21]: y_pred = CNN_model.predict(i
         #get_probability(y_pred)
         Y_pred_classes = np.argmax(y
         Y_predicted = Y_pred_classes
         classified_as = map_characte
         print("Classification:", cla

         Classification: B
```

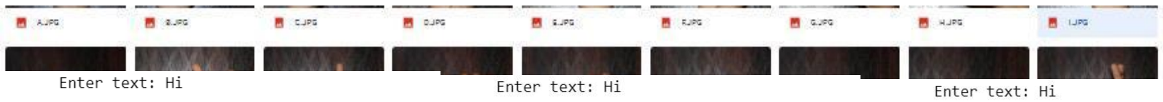


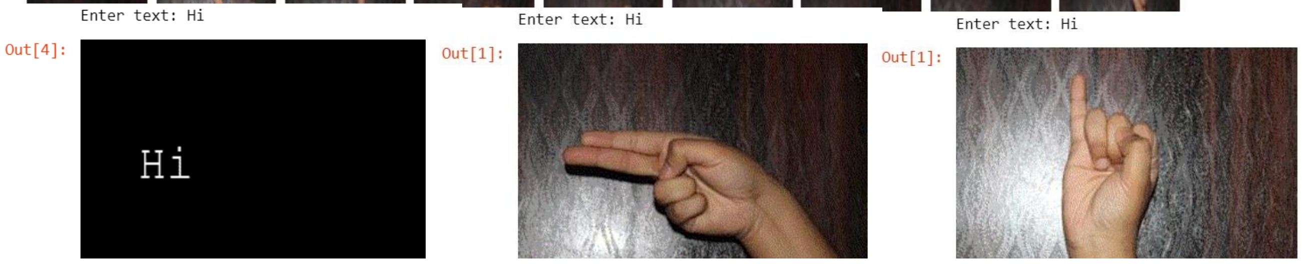
```
Out[46]:

In [47]: y_pred = CNN_model.predict(i
         Y_pred_classes = np.argmax(y
         Y_predicted = Y_pred_classes
         classified_as = map_characte
         print("Classification:", cla

         Classification: G
```

Figure 8: Output from ASL recognition system

4.2 Results for text to ASL finger spelling system

Figure 9: ASL alphabets images





```
Enter text: Hi
Out[4]:
```

```
Enter text: Hi
Out[1]:
```

```
Enter text: Hi
Out[1]:
```

Figure 10: Output GIF from left to right when input is "Hi"

5. Conclusion

Hand gestures are a powerful way for human communication, with lots of potential applications in the area of human computer interaction. Vision-based hand gesture recognition techniques have many proven advantages compared with traditional devices. However, hand gesture recognition is a difficult problem and the current work is only a small contribution towards achieving the results needed in the field of sign language recognition. This paper presented a vision-based system able to interpret hand gestures from the American Sign Language and convert them to text or speech. After that we also did the opposite. We have been able to convert text to sign language. The proposed solution was tested in real time situations, were it was possible to prove that obtained classification models were able to recognize all the trained gestures being at the same time user independent, important requirements for this type of systems. The selected hand features, in conjunction with machine learning algorithms, proved to be very efficient, allowing their application in any real-time sign language recognition systems. As future work it is intended to keep improving the system and make experiments with complete language datasets. As a final conclusion one can say that although there is still much to do in the area, the proposed solution is a solid foundation for the development of any vision-based sign language recognition user interface system. The sign language grammar can be easily changed and the system configured to train the new language gestures.

Journal of Innovative Image Processing (JIIP) (2020)
Vol.02/ No. 02
Pages: 65-76
https://www.irojournals.com/iroiip/
DOI: https://doi.org/10.36548/jiip.2020.2.001

References

[1] Pavlovic, V, Sharma, R., &Huang T. (1997). Visual Interpretation of Hand Gestures for Human-Computer Interaction (HCI): A Review. IEEE TOPAMI, VOL. 19, NO. 7.

[2] Jiyong, M.W., Gao, Jiangqin, &Wu Chunli, Wang (2000). A Continuous Chinese sign language recognition system in Automatic Face and Gesture Recognition. Fourth IEEE International Conference on 2000.

[3] Vogler, C.M., &Dimitris (2004). Handshapes and Movements: Multiple-Channel American Sign Language Recognition Gesture-Based Communication in Human-Computer Interaction. Ed. A.V.Camurri, Gualtiero. Vol. 2915: Springer Berlin Heidelberg. 247-258.

[4] Gunasekaran, K., &Manikandan, R. (2013). Sign Language to Speech Translation System Using PIC Microcontroller. International Journal of Engineering and Technology (IJET), Vol 5 No 2.

[5] [5] Kalidolda, N., &Sandygulova, A. (2018). Towards Interpreting Robotic System for Finger spelling Recognition in Real-Time. HRI Companion: 2018 ACM/IEEE International Conference on Human-Robot Interaction Companion.

[6] [6] Liang, R., & Ouhyoung, M. (1998). Real-time continuous gesture recognition system for sign language. Proc Third. IEEE International Conf: on Automatic Face and Gesture Recognition, pp. 558-567.

[7] [7] LeCun, Yann, Lon, Bottou, Yoshua, Bengio, & Haffner, Patrick (1998). Gradient-based learning applied to document recognition', Proceedings of the IEEE 86, vol. 11, pages 22782324.

[8] [8] Simonyan, Karen, & Zisserman, Andrew(2015). Very Deep.... Recognition. Conference paper ICLR.

Authors Biography

The paper presented above is a part of major project by 4th year student at Pulchowk Campus, IOE. They belong to the Department of Electronics and Communication Engineering. The project was successfully supervised by Prof. Dr. Subarna Shakya who is a professor at Pulchowk Campus. Through there major project, they thought of helping the society in technical way. With this idea, they came across the concept of helping the deaf and dump person to be able to convey their message to everyone is their own way. The supervisor was helpful enough throughout this journey of making the concept turn into application which may help the targeted audience in some way.

Amrita Thakur

Shirish Shrestha

Pujan Budhathoki

Sarmila Upreti

Prof. Dr. Subarna Shakya

SUBARNA SHAKYA has received the MSc and PhD degrees in Computer Engineering from the Lviv Polytechnic National University, Ukraine, 1996 and 2000 respectively. He is the Professor of Computer Engineering, Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering, Pulchowk, Tribhuvan University, Nepal. **He has served** as Executive Director at National Information Technology Centre, Government of Nepal.
His areas of research interests include E-Government system, Computer Systems & Simulation, Cloud computing & security, Software Engineering & Information System, Computer Architecture, Multimedia system.