Acta Scientific COMPUTER SCIENCES

Volume 4 Issue 10 October 2022

Research Article

An LSTM Based Sign Language Recognition Model in Support of Patients with Hearing Disabilities

Ashwin Krishna Raparthy[1], Bhim Prajapati[1], Santhi Priya Yalla[1], Vigneshkkar Ravichandran[1], Enrico De Santis[2] and Parisa Naraei[3]*

[1]Postgraduate AI Student, Lambton College, Canada

[2]Machine Learning Engineer and Chief Technology Officer, SisterPomos and Scientific Researcher, Sapienza Università di Roma, Italy

[3]Professor, Cestar College of Business, Health and Technology, Department of Artificial Intelligence, Lambton College, Applied Researcher, Toronto Metropolitan University Alumni, Canada

***Corresponding Author:** Parisa Naraei, Professor, Cestar College of Business, Health and Technology, Department of Artificial Intelligence, Lambton College, Applied Researcher, Toronto Metropolitan University Alumni, Canada.

Received: July 26, 2022

Published: September 23, 2022

Abstract

Human beings are wonderful creations of nature. People are known for the intelligence they have. Among the various creations of human thoughts, language is also one that human beings use to communicate with each other. But for, differently abled people who cannot hear have a different language to communicate with, called sign language. This type of language is expressed using the physical articulation of the body parts. In this case, it should not be confused with the body language that normal people use. Deaf people primarily use sign language, but people who cannot speak also use this language.

With the evolution of technology and advancement in AI, computers can now process human language using NLP. In this project, we use a deep learning method to predict sign language. We are building a web app through which a sign user can interact with a normal person who cannot understand sign language. We have used a convolutional neural network (CNN) to get the key points of the human body like the face, shoulder, arm, hand, and fingers. Then we used long short-term memory (LSTM) to model the neural network. Our model can predict sign language with an accuracy of 95.8%.

Keywords: Convolutional Neural Network (CNN); Hidden Markov Model (HMM); Artificial Neural Network (ANN)

Introduction

Language is a medium that helps us to communicate with each other. Sign language is also a form of language used by deaf people or people who cannot speak to communicate with each other. Sign language complicates the communication between the sign user community and non-sign users. There are other alternatives for communicating with sign and non- sign users, like writing down the conversation, but there is a problem with this method mainly because deaf people are less skilled at writing, and this method will be a slow process that will not be applicable in an emergency situation. For example, it will be necessary to communicate quickly when an accident occurs. So, with these limitations, there have been some innovations in sign language to take care of the situation.

Related work

There have been many innovations in this field to break down the barrier between sign and non-sign users. Thad Starner used one camera to track the hands. They track the hand by color. They use solidly colored gloves with accelerometers or natural skin tones to track the hands. So, they produced the shape, orientation, and trajectory of the hands, passing them to the Hidden Markov model (HMM) to get the sign's meaning.

Similarly, Zahoor Zafrulla [1] used Kinect, a depth mapping camera for sign language recognition. They have done image processing to get the position of the hands. They extract face and hand regions using their skin colors, compute the blobs and track the location of each hand using the Kalman filter and draw the trajectories based on the hand motion and predict the sign based on the trajectories.

Our work is more related to the work done by Lionel Pigou [2]. They have extracted the features from the frame sequence. It will represent a vector consisting of one or more feature vectors called descriptors. They have used Convolution Neural Network (CNN) for feature extraction and Artificial Neural Network (ANN) for classifying the data. They got a cross-validation accuracy of 91.7%.

Dongxu Li [3] presented vision-based sign language recognition help deaf people communicate with others. There are two tasks related to sign language recognition. They are word-level sign language recognition and sentence-level sign language recognition. There are different models for this purpose; a Holistic visual appearance-based approach, a 2D human pose-based approach, and a Pose-based temporal graph convolution networks (pose-TGCN). Pose-TGCN further boosts the performance of the pose-based approach. They got 62.63% accuracy in testing the model. There can be challenges in the accuracy because subtle differences in body motion and head poses may translate into different meanings, as presented in figure 1.

Figure 1: Same sign representation for words "Rice" (top) and "soup" (bottom) (Dongxu Li, 2020)

Figure 2: The verb "Wish" (top) and the adjective "hungry" (bottom) correspond to the same sign (Dongxu Li, 2020).

The dataset consists of 21,083 videos performed by 119 signers, and at least three different signers perform each sign to bring dynamism to the dataset. A classification algorithm like SVM labels the signs with the corresponding words, then uses CNN to extract the holistic features from image frames and use the extracted features for classification. Their approach was first to extract body key points and then concatenate their location as a feature vector. Then the extracted features are then fed to a stacked GRU for recognizing signs. Some employ 3D CNNs instead of encoding spatial and temporal information separately. And hire a face detector and face embedding provided by Face Net and compare Euclidean distances among face embeddings to confirm the diversity of the signers.

The temporal boundary is an indicator that indicates the start and end frames of a sign. They created a body bounding box to reduce side effects caused by background and let models focus on signers. So, they used YOLOv3 as a person detection tool to identify body bounding boxes of signers in the video. They used 2D CNN to extract spatial features of input images while RNN to capture long-term temporal dependencies among inputs, and training of the model is done with an Adam optimizer. So, the problem with the data collection part is that understanding sign language requires specific domain knowledge. Hence, labeling many samples per class is unaffordable.

Methodology

Our approach includes various steps such as data collection, data cleaning, data analysis, and model building and deployment to achieve the desired result, like recognizing the sign language words.

Project architecture

Below is the architecture of our overall project demonstrated in figure 3.

Figure 3: Solution Architecture.

Figure 3 is the architecture designed to present how the different components and APIs are connected. Here, the docker open-source containerization platform has been adopted for docking the UI, backend, and MongoDB interface. All instances have been deployed on Amazon EC2. These instances are stored in the shared EBS cloud storage of Amazon. When a user logs in using the URL, the docker image will load into the computer. The user communicates through Rest API and hits the configured Nginx server reverse proxy. The proxy server determines what to give to the browser. Rest API does the API calls to the UI, backend, and MongoDB. MongoDB is at the lower end of the architecture.

The backend is backed in the local system, where the training and testing of the model are done. So, this is also stored in the cloud through the reverse proxy server via the MongoDB connection. Whenever a new user logs in to the system, they connect to the system through the reverse proxy server. It provides a gateway between users and the internet. It prevents others from entering the private network. Like this, all the communication between the two users happens through a peer-to-peer connection. The connection is based on the WebRTC protocol. So, this peer-to-peer connection will help in the performance and privacy of the information transferred.

Dataset collection and analysis

The designed solution is based on AWSL (American World Sign Language), therefore the first task was to search for the dataset. The source of the dataset is YouTube video in a JSON file that consists of the information of the dataset like the URL, start frame, end frame, video ID, etc.

After getting all the videos and JSON files that consist of the information of the dataset like the URL, start frame, end frame and video ID, it was evident that there was a varying number of samples for different words, which will introduce biases in results for detecting all the signs correctly and after checking the video consistency for each word. There is a lot of difference in the total number of each word video which may cause bias in our model. To address it, we first selected the most familiar samples with a high number of videos and analyzed them. Even though we choose a few words, the frames count for these words is different, which causes inconsistent ending video frames.

Due to this inconsistency in the available data, we had to opt for another data, i.e., setting up our dataset. We researched different deep learning models to select the best model to check the accuracy of the words. In this process, the LSTM model was selected as it best suits the requirements of the project.

Modeling

LSTM has been selected as the learning algorithm for the model training because we have to find the dependency in the sequence of critical points to predict the sign words. LSTM is suitable for the time-series data. We have used CNN to extract the key points elements of the face and hands.

Figure 4: LSTM Architecture (https://blog.floydhub.com).

LSTM is a type of RNN where the network is trained through a sequence of inputs across time. It has short-term memory that stores the state of the last cell (hidden state) and long-term memory that holds the state of the current cell (cell state). It has three gates; an input gate, an output gate, and forget gate. The cell then uses the gate to regulate the information to be kept or discarded at each time step before passing the short-term and long-term information to the next cell. The input gate defines what added information will be stored in long-term memory by taking the lead from current input and short-term memory from the previous step. The forget gate defines if the information in long-term memory will be kept or discarded. The output gate will take the information from current input, previous short-term memory, and currently generated long-term memory and produce new short-term memory, which will pass in the next step.

Modeling

We are using the key points to train the model. We have extracted the key points using the CNN model. The key points include the face marks, shoulder, arms, hands, and fingers. So, in each video, it will capture the 40 frames. As a first step, we built a dataset with three words, namely 'hi, 'I", and 'name' signs, and stored that data in MongoDB. We used a label map to create a dictionary of label names to its id. We trained our model with 2000 epochs. At epoch number 85, our model has 100% epoch_categorical_accuracy with '0' epoch_loss. So, we stuck to this model and further collected more datasets to train it on different sign languages. For this project, we have created a user interface using React library from which we can prepare our dataset and test our model. Collectively, ten sign videos for 13 selected words is made individually and the data was stored in the created database. Doing so, each word has 40 sign videos to train and test the model. Below is the overall model summary of our project as demonstrated in figure 5.

Figure 5: Model summary.

The neural network model comprises an LSTM layer with 100 units and an input size defined as (40, 1662), equal to the number of frames per video and length of extracted key points. We added the dropout layer to avoid the overfitting to data. Similarly, we said the first dense layer with Relu as activation function and the next dense layer with 'SoftMax' as activation function. We then compiled the model with Adam optimizer.

Results and Findings

After building the model, we trained the model up to 839 epochs, where we got an accuracy of 93.64% in the training set, and an accuracy of 95.83% in the test set. Also, we plotted the confusion matrix to see the model's performance. By observing the values of true positive, true negative, false positive, and false negative, the model is stable predicting and ready for deployment.

Tensor board analysis

Accuracy has been increasing for every epoch, and we have got satisfactory accuracy at 840th Epoch as demontsrated in figure 6 and loss has been decreasing substantially concerning each epoch as demonstraed in figure 7.

Figure 6: Accuracy in training data concerning increase in the epoch.

Figure 7: Loss in training data concerning increase in the epoch.

Web application

We developed an application that will consume our model and make a prediction of sign language between two or more people on a real-time basis and record the data. Figure 8 demonstartes a snippet of the User Interface.

Figure 8: User Interface for interaction with backend and data collection.

This application consists of 'join room' and 'record dataset' as the primary features, as shown in the above figure. While 'join room' allows the user to join the room with a specific username and room id, 'record dataset' collects the data for videos. Here people can log in with a unique room id and username. The user can either log in as a sign user or a regular user. If they log in assigned user, then they can make sign language gestures, and the corresponding word prediction will be displayed in another user profile. If they logged in as a regular user, they will not be able to make sign language gestures but only see the predicted words from other sign language users.

So, we have used Rest API to make API call to the UI and configured it in the Nginx server reverse proxy.

Once the users log in, they will establish a peer-to-peer connection using a WebRTC proxy. It will use the IP address of connecting peers to transfer the real-time data. We purchased a domain name and configured it with our system so that every user can access this application using the URL. Finally, we connected our web app to our domain name(loud-signers.com).

Conclusions and Future Work

In a nutshell, this paper presented the whole process of developing a state of the art model to predict the sign language with high accuracy. It is a proof of concept to help break down the barrier between the sign user community and the non-sign user community. As part of the future work, the system can be open sourced so that people around the globe can contribute to the project and use the app easily to communicate with sign users.

Contributions (Services)

- Application: https://loud-signers.com/
- Repository Front end: https://github.com/Vigneshkkar/Sign-Language-Detection
- Repository Backend: https://github.com/Vigneshkkar/Sign-Language_detection_Backend
- Docker Image Front End: https://hub.docker.com/repository/docker/vigneshkkar/slr-ui
- Docker Image Backend: https://hub.docker.com/repository/docker/vigneshkkar/slr-backend

Bibliography

1. Zafrulla Z., *et al*. "American sign language recognition with the kinect". In Proceedings of the 13th international conference on multimodal interfaces (2011): 279-286.

2. Pigou L., *et al*. "Sign language recognition using convolutional neural networks". In European conference on computer vision (2014): 572-578.

3. Li D., *et al*. "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison". In Proceedings of the IEEE/CVF winter conference on applications of computer vision (2020): 1459-1469.