Real-Time Sign Language Detection for Video Conferencing

Team 28: Tyro Soh (A0216134R), Vikas Harlani (A0214360U), Zhuang Jianning (A0214561M), B. Nishanth (A0216347E), Low Junwei (A0216084J)

National University of Singapore
School of Computing

Abstract

Existing video conferencing applications such as Zoom are lacking in accessibility features for sign language users, who are currently restricted mostly to typing for communication. Our paper aims to develop models capable of detecting sign language in real-time and translating it into speech/captions quickly and with high accuracy, which will allow sign language users to communicate by signing on video live. These models were created using a combination of Convolutional Neural Networks (CNN), Long Short-term Memory (LSTM), and the open-source MediaPipe Holistic framework, which were then assessed on speed and accuracy by an extensive dataset of 700 videos. We justify our final selection of an LSTM model that utilises the MediaPipe Holistic framework and discuss several possible avenues of improvement that could be made to our chosen model to further enhance the prediction accuracy.

Introduction

During the Covid-19 pandemic, there has been a meteoric rise in the use of video conferencing tools.[1] As a result of social distancing, applications such as Zoom, Microsoft Teams and Google Meet have become essential for remote work and for connecting with loved ones. Even as lockdowns and restrictions begin to lift, video conferencing has remained integrated into many parts of our lives due to the flexibility, convenience, and productivity it brings.[2]

Although video conferencing aims to help connect people, for those who rely on sign language due to hearing or speech disabilities, using these applications to communicate may instead feel inaccessible. During video meetings, applications have algorithms to auto detect and highlight the active speaker. In a grid layout, a yellow square will surround the speaker's thumbnail. In a full-screen layout, a large view of the active speaker would be shown. However, when a person is signing, the software does not give them any special recognition, making it difficult for signers to gain the floor and for others to tell who is signing at any given moment.[3]

There are alternative ways to get around the problem, such as by using built-in chat or reaction features. However, these channels can be easily missed, ignored or even burden other participants with a cognitive load to frequently check if someone is signing. Even if the message is noticed, the signer is still not automatically highlighted or given the active speaker view. Moreover, using typing as the main form of communication may not be as natural or efficient for those who rely on signing. Instead, having a real-time sign language translator will allow those more comfortable with signing communicate naturally in real time.

Recognition and classification of image and videos is a fundamental challenge of computer vision. Although there has been increased interest and progress in utilising machine learning techniques for detecting and recognising sign language, there has yet to be much integration into video conferencing solutions. (Adeyanju et al., 2021) Furthermore, there is no single sign language used around the world, with variations naturally developing through different groups of people interacting.[4] As expected, applications for Singapore Sign Language (SgSL) are further limited.

In our project, we examine and compare the performance of various machine learning models in detecting and translating SgSL. We also propose an application for video conferencing and demonstrate its feasibility though the use of a webcam.

Proposed Application

In order to make virtual communication more accessible for those who rely on sign language, we propose a real time sign language detection and translation system that is integrated with Zoom/other video conferencing applications. For our application, we focus on translations for SgSL as there is a lack of application for it despite being highly relevant in our local community.

Our first feature is the ability to detect when a user is signing. By enabling an option to detect sign language, our system will monitor the video feed whilst the signer is unmuted. Upon exceeding a certain threshold of confidence for any

[1] https://www.bbc.com/news/business-56247489
[2] https://getvoip.com/blog/2020/07/07/video-conferencing-stats/
[3] https://www.popularmechanics.com/technology/design/a28981613/google-ai-developing-asl-detection-video-calls

[4] https://www.ai-media.tv/ai-media-blog/sign-language-alphabets-from-around-the-world/

sign, the system will highlight the user as the active speaker. This can be done by sending a 20 kHz tone which mimics a person speaking whilst not being in the audible frequency range. This works with any video conferencing application.

Our second feature is accurately translating a user's signs in real time. After detecting that a user is signing, our system will begin translating them. With the help of the video conferencing application's transcribe feature or text-to-speech feature, other users in the meeting will be able to read/listen to the user signing. As a result, signers can converse more naturally with others in real time.

Currently, signers will be able to run our Python program[5], which captures their webcam output, and opens a separate window translating the words they are signing, in real time. With a high level of accuracy, our system shows to the users the SgSL signs that it has detected. Figure 1 below shows an example of such an output.

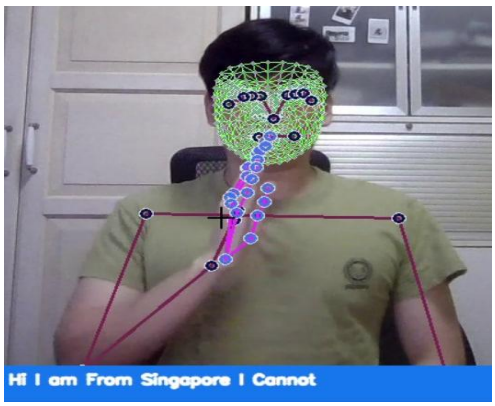

Figure 1: Our program outputting user's signs

Application Characteristics/Requirements

Multi-Class Classification

For this study, there are 14 possible class labels due to there being 14 words in our sentence. Each word is assigned a probability, and the final predicted word will be the word with the highest assigned probability. The chosen word's probability must exceed a certain threshold before it is selected. This helps ensure that random actions that do not constitute a signed word will not wrongly generate words.

Spatial and Temporal Components

SgSL is a visual form of communication that requires hand sign movements. The inputs are frames from a video/feed and not just single images or variables. The frames being a 2D image, each cell is related to neighbouring cells. Thus, the features would have a spatial component that we need to extract. The frames being sequential forming a video makes each frame related to neighbouring frames. Thus, the features would also have a temporal component that we need to extract.

Requires Fast Prediction Time

To keep up with the user signing, our application must be able to predict the signed word fast enough such that it is able to output the transcribed words in real time. This ensures the generated captions/speech is synchronised with the signing user, which will improve other listeners' ability to understand the signer.

Requires High Accuracy

High accuracy is necessary to ensure the predicted words are aligned with what the user is signing. This is especially important in ensuring the sentences generated are cogent and intelligible. An incorrect word in the middle of a sentence could distort or misrepresent what the signing user is trying to communicate, which will make it difficult for listeners to interpret the message.

Extensive Dataset

For our purposes, the dataset would have to contain multiple videos of different people signing the same word. This is necessary for the system to effectively be able to recognise the same word, even when performed by different individuals. Online datasets for SgSL words rarely had multiple videos of the same word.

Machine Learning Techniques

Convolutional Neural Networks (CNN)

Convolutional Neural Network is a type of neural network that is commonly utilized for classification tasks and computer vision tasks[6]. Leveraging algebraic concepts such as matrix multiplication, this special type of neural network has become the state of the art for visual applications. The layers of this network identify patterns among images, making it highly likely to be useful in dealing with the spatial component of sign language detection and translation, with the frames of the video being the images.

There are 3 main type of layers in CNNs: Convolution layer, Pooling layer and Fully-Connected layer. Earlier layers of the network aim to process simpler features such as possible edges and colors, with the complexity and detail of image processing increasing with each layer.

The first layer type is the convolutional layer which plays a vital role in the network. The input data for this layer is the image, a matrix of pixels in 3D. The three dimensions being the height, width, and depth (number of color channels) for

[5] https://github.com/jianningzhuang/CS3244-Sign-Language-Detection

[6] https://www.ibm.com/cloud/learn/convolutional-neural-networks

the image. The input can also be 4D if multiple images are fed to the network. Then, a filter which is typically a 3x3 matrix shifts sequentially through the image from top left to bottom right. calculating the dot products of itself and the part of the image currently being checked. These results are collected in an output array as seen in the figure. Many of such convolutional layers stacked together are capable of picking up on patterns such as lines, gradients, and circles in the image.

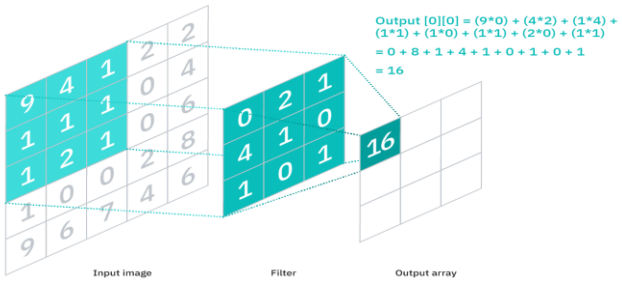

Fig 2: Depiction of filter in Convolutional Layers

Next, pooling layers are used to reduce the parameters of inputs from previous layers. This step greatly reduces complexity and the risks of overfitting. Finally, the Fully-Connected layer, performs the classification based on the features extracted from previous layers and their filters. Convolutional and pooling layers often use ReLu functions, while the Fully-Connected layers tend to rely on a SoftMax activation function for classification, which returns probabilities between 0 and 1.

Long Short-term Memory (LSTM)

Recurrent Neural Networks (RNN) exhibit temporal dynamic behaviour by maintaining a hidden state that is updated at each time step as a sequence of data is processed. This hidden state is defined as a function of the current input and previous state as shown by the following recurrence relations:

$$h_t = \sigma_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = \sigma_v(W_{hv}h_t + b_v)$$

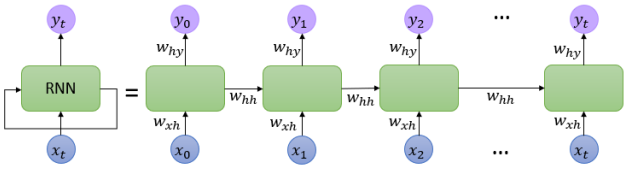

Figure 3: Rolled vs unrolled RNN

where x_t is the input vector, h_t is the hidden state vector, y_t is the output vector, W is the weights matrix, b is the bias and σ is activation function.

Although RNNs have been applicable to sequential tasks such as handwriting or speech recognition, they may face problems such as vanishing or exploding gradients which affects their ability to learn long term dynamics. As the gradients are back propagated through time via increasing layers in the recurrent network, many factors of W_{hh} with weight values that are less than 1 or greater than 1 will result in vanishing or exploding gradients respectively. In the case of vanishing gradients, errors further back in time have smaller gradients which would bias parameters that capture short-term dependencies.[7]

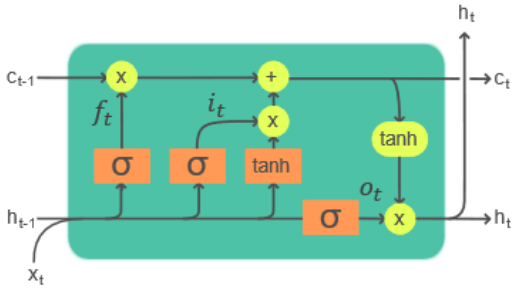

Figure 4: LSTM unit with cells and gates

LSTM overcomes the vanishing gradient issue by incorporating cell units and gates that allow them to learn when to forget previous hidden states and when to update them given new inputs. The functions of each gate are as follows:

To forget irrelevant parts of the previous state:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

To store relevant new information into the cell state:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

To selectively update cell state values:

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t$$

To control what information is sent to the next time step:

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$h_t = o_t \circ \tanh(c_t)$$

These additional cells and gates allow LSTM to learn complex and long-term temporal dynamics, making it well suited for making classifications and predictions based on time series data. This is especially applicable in dealing with the temporal nature of sign language detection and translation, where sequences of frames in a video are used to classify different signs.

Chosen Models for Experimentation

Previous works generally take one of two approaches to reduce the network's parameters: (1) using pose estimation on the original videos; or (2) using CNNs to get a feature vector per frame.

[7] http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf

ConvLSTM

ConvLSTM is a type of recurrent neural network for spatio-temporal prediction that has convolutional structures in both the input-to-state and state-to-state transitions (X. Shi et al., 2015). It is a recurrent layer just like in LSTM, but internal matrix multiplications are exchanged with convolutional operations. This allows inputs to the layers having a 2D/3D dimension instead of 1D vector input.

For our model we use ConvLSTM2D layers from the Keras package in Tensorflow.

CNN + LSTM (LRCN)

The structure of this architecture is to use CNN and LSTM[8] networks, where the CNN is used to extract complex features from images, and LSTM is used as a classifier. Each frame is passed into CNN layers to extract the spatial features and the extracted features are then passed into the LSTM layers to further extract the temporal features.

For our model we use Conv2D, MaxPooling2D and LSTM layers from Keras package in Tensorflow.

MediaPipe Holistic + LSTM (MP + LSTM)

MediaPipe Holistic is a pose estimator that extracts landmark features for the hands (21 landmarks per hand), pose (33 landmarks) and face (468 landmarks) in real-time from each frame of a video.[9] Since sign languages are visual languages produced by the movement of the hands, face and body, pose estimation is suitable as a lower dimensionality representation of sign language videos. Such a representation is also background and person independent, able to protect privacy and focuses on the recognition of motions which allows it to be generalised to unseen videos. After extracting the landmark values using MediaPipe Holistic, the spatial features are then passed into the LSTM layers to further extract the temporal features.

For our model, we use the Holistic model from MediaPipe and LSTM layers from Keras package in Tensorflow.

Experiment Methodology

Dataset

To train our models for use in Singapore, we collected videos of 14 different words in SgSL that make up the sentence. “Hi, I am from Singapore. Paiseh, I cannot speak. But machine learning can help translate!” The signs were taken from documentation in the SgSL Sign Bank.[10] Our dataset comprised of 700 videos, with each word having 50 individual videos performed by 5 different signers.

Processing of Data

For the ConvLSTM and LRCN models, the inputs would take in the frames of the video. The videos were processed to 30 frames, each frame resized to (128,128) and containing 3 channels RGB. This meant that the input shape for each input instance was (30,128,128,3) for these 2 models.

For the MP + LSTM model, we first ran the MediaPipe pose detection model on the video to abstract the important human body keypoints for each frame. We collected 1662 keypoints for each frame. The input shape for each input instance was (30,1662) for this model.

Hyperparameters

The problem being multi-class classification, the aim is to classify the video by choosing the most probable class. The Softmax activation function is thus used as the output layer activation function. Softmax function converts[11] a vector of numbers into a vector of probabilities that sum to 1. The probabilities of each value are proportional to the relative scale of each value in the vector. This allows the models to predict probabilities. The loss function is then also set to Categorical Cross-entropy. Cross-entropy builds upon the idea of entropy from information theory and is used to evaluate the loss of probabilities. It calculates the sum of separate loss for each class label per observation.

There was many other hyperparameters to select such as, learning rate, number of layers, number of nodes in a layer, activation function, batch size and optimizer. The many combinations of these hyperparameters were trialed during validation for tuning.

To prevent overfitting the data, the training and validation loss were monitored and the best weights were restored.

Validation

K-fold Cross Validation was used to validate each model, comparing between different combinations of hyperparameters to train a model, and also evaluating their performance compared to other models.

The training data was randomly split into 5 folds. For each iteration, 1 fold was left out as the test set and the remaining 4 folds were used to train the models. The remaining 4 folds were further split into 0.8/0.2 (training/validation). Validation set was not utilized in the training of the model, and instead used to monitor the validation loss, helping to prevent overfitting and selecting the best weights. The trained model was then tested on the test set left out to obtain the categorical accuracy, precision and recall from that iteration.

[8] https://www.sciencedirect.com/science/article/pii/S2352914820305621
[9] https://google.github.io/mediapipe/solutions/holistic.html
[10] https://www.sgslsignbank.org.sg/
[11] https://machinelearningmastery.com/softmax-activation-function-with-python/

This is done for all 5 folds individually being left out and the mean scores of 5 iterations is then recorded.

Performance metrics

The problem being multi-class classification, categorical accuracy and F1 score was used to evaluate the performance. Categorical accuracy calculates how often the predictions match the one-hot labels, it is calculated by $\frac{Accurately\ predicted\ instances}{Total\ instances}$.

The F1 score is another metric that measures accuracy. The F1 score used was calculated using macro averaging, calculating the F1 score for each class label and get the average of all the scores.

First, we obtain the individual precision and recall for the class. In the case of multi-class classification, we would consider the precision and recalls for each class from the categorical confusion matrix using the following,

$$\Pr e\,cision_A = \frac{Correct\ Classification\ of\ Class\ A}{Total\ Classifications\ as\ Class\ A}$$

$$Recall_A = \frac{Correct\ Classification\ of\ Class\ A}{Total\ Instances\ of\ Class\ A}$$

We then get the F1 score for the class label using,

$$F1\_Score_A = \frac{(2*Precision_A*Recall_A)}{Precision_A+Recall_A}\text{ma}$$

We then average the individual class F1 scores to get the macro averaged F1 score.

$$F1\_Score = Mean(F1\_Score_A, \dots, F1\_Score_N)$$

Experimental Results and Analysis

Model	Categorical Accuracy	F1 Score	Prediction Time
MP + LSTM	0.961	0.962	0.000951s
ConvLSTM	0.959	0.954	0.195s
LRCN	0.973	0.972	0.125s

Table 1: Results obtained

In choosing our model in the context of real time sign language translation, accuracy of prediction and prediction times must be considered. From our results there is a trade-off between higher accuracy vs faster prediction time. Considering faster prediction times would improve the smoothness of the translation and imply a lightweight model that interferes less with call quality, we determined that the MP + LSTM model would better suit our needs. Considering also that there are possible modifications to improve accuracy that is discussed in a later section.

Further Analysis on selected MediaPipe Holistic + LSTM Model

MediaPipe Holistic using pose estimation results in lower dimensionality representation of frames compared to features extraction using convolutional layers. This allows the model to be faster and lightweight. The loss in information that pose estimation does not provide that is present in CNN models does not seem to affect the accuracy significantly.

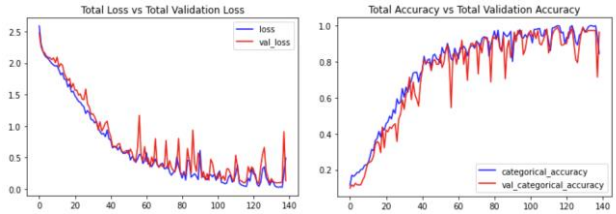

Figure 5: Training loss/validation accuracy

Analysis of the individual class confusion matrix shows that most classes were able to be predicted correctly for all the test data. The error then comes from the model failing to predict “translate” signs and predicting it as “machine” or “help” instead. Looking at the signs we realise that these signs were very compact compared to other signs, with many movements happening with the hands together. This could be a disadvantage of using pose estimation, when the hand to hand or hand to face interactions potentially cover up some landmarks.

The model was tested real-time and works well (Fig 1). The predictions are fast, and the accuracy was satisfactory enough with most words in the sentence predicted correctly.

Challenges faced

Limited Dataset

Due to the lack of datasets for SgSL, we were limited to five different signers performing the same signs multiple times to make up the training dataset. This is instead of having unique signers perform each sign just once. A potential problem of having such a limited dataset is overfitting, which occurs when the model fits too closely with the training data (Hawkins 2004). To avoid overfitting, we monitored the training/validation loss and had set a threshold value to stop training at.

Lengthy Training Time

For each of our models, there were considerably lengthy training times: upwards of 10 minutes for each epoch, and cross validation on ConvLSTM taking more than 12 hours. As a result, there was a trade-off between the number of videos in the dataset and total training time, which led to our final choice of dataset.

Modifications and Improvements

Non-signing Gestures

To more accurately detect when a user has started or stopped signing while unmuted, the problem of sign language detec-

tion and sign language translation could be focused on separately. We could train another MP + LSTM model for sign language detection with a separate dataset that comprises of instances labelled signing and instances labelled non-signing. Negative examples and distractors could include videos of people staying still, accidental movements or gestures that are not signs. After determining that a person has started signing, our original model can be used to classify the detected sign. This might improve the confidence of our application in identifying actual signs and prevent unintentional spotlighting of signers.

Personalized Predictions

The MP + LSTM model above was trained using signs from various signers, so that a variety of users can use the final model. Once the model is deployed and integrated into the conferencing application, the model can further train itself to be personalized to the user of the application. However, this would require some feedback from the user to let the system know if wrong translations were made during calls. Since signing behaviours vary from person to person, having a bespoke model that caters to a single user should highly increase prediction accuracy. While the collection of personal data does raise ethical concerns, our model of choice only collects positions of keypoints instead of the RGB images themselves, which mitigates personal privacy concerns.

Predict Future Words

LSTM or other pre-trained natural language processing models such as BERT can further be used to predict upcoming words in a sentence. By narrowing down the potential words that can appear next, our model can be more confident about its predictions and reduce the chance of irrelevant words being accidentally translated. For example, 'learning' and 'trampoline' may have high prediction probabilities due to their signs looking similar. However, if the preceding word was 'machine', a model trained to know sequential patterns of words will output 'learning'. The Next Word Prediction can also be dynamically adjusted for different users based on their speech patterns, not unlike autocorrect features on mobile devices.

Conclusion

In this paper, we suggest three possible models for predicting sign language with video footage in real-time and evaluated them based on their categorical accuracy and prediction speed. We found that the LRCN model achieves a marginally higher prediction accuracy when compared to the ConvLSTM and MP + LSTM models, but the MP + LSTM model is both light-weight, and more capable of keeping up with live signers to output captions/speech at a suitable pace for listeners to understand. We also addressed possible issues with our existing models, and proposed possible improvements that could mitigate these limitations, such as personalized next word predictions, and more safeguarding against false detection of signing gestures.

Member Roles

This study was the result of the combined effort of all 5 team members. The group met regularly to update every individual's progress and to discuss how to advance further. We are satisfied that everyone carried out their role properly. Jianning worked on research for LSTM and explored potential models for MP + LSTM. He has gained a better understanding on the technical details of how RNNs model temporal behaviour and how LSTM can overcome problems like vanishing gradient with additional cells and gates. Tyro worked on the ConvLSTM model and researching possible limitations and extensions of the models. He has become more aware of potential ethical issues machine learning researchers face, such as the possibility that automated sign language translation could replace professional human translators. Nishanth researched on CNN and its potential models. He realised extensive use of linear algebra in the layers could result in identifying patterns from images. Vikas worked on forming the application features as well as the MP + LSTM model. He has gained insight into the challenges faced by those who rely on sign language and ways to solve them. Junwei worked on processing the data as well as on the LRCN and ConvLSTM models. From researching and trialing the effects of the many hyperparameters, he has learned more on how each of them affects the model and training.

References

Cox, M. T. Perpetual Self-Aware Cognitive Agents. *AI Magazine* 28(1): 32–45, 2007

Hawkins, D. M. The Problem of Overfitting. Journal of Chemical Information and Computer Scientists 44(1): 1-12., 2004

J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell "Long-term Recurrent Convolutional Networks for Visual Recognition and Description", 2016.

A. Soleimany, "Deep Sequence Modeling." 24-Jan-2022.

I. A. Adeyanju, O. O. Bello, and M. A. Adegboye, "Machine learning methods for sign language recognition: A critical review and analysis,"

Xingjian Shi Zhourong Chen Hao Wang Dit-Yan Yeung: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, 2015