A Sign Language Translation System Based on Deep Learning

Siming He

Abstract **– lingual-disabled people around the world suffer from communication problems and discrimination (World Federation of the Deaf, 2016). Even worse, according to the World Health Organization, in 93 counties, 31 countries didn't have sign language interpreting services, while 30 countries had 20 or fewer qualified interpreters (World Health Organization, 2018). Due to this issue, I hope that software of sign language translation, instead of interpreters, could be more efficient and reliable at helping deaf or mute. After research, I expect to design a translation system based on deep learning enabling videos to be recognized and translated into speech. My design successfully achieves this expectation. In my design, OpenPose, an open source convolutional neural network (CNN), is used to estimate hand pose of each frame from videos. Then, I build a long short-term memory (LSTM) network under Tensorflow framework to classify patterns of hand pose. To train this network, a dataset containing 3060 videos of seventeen different sign language words or sentence is also created and augmented. After attempts of optimization, the LSTM Network reaches 93.62% accuracy and the function of sign language translation into speech is realized. In addition, the success of using 2D raw videos raises possibilities of real-time sign language translation on phones whereby sign language translation system is popularized. What to improve next is to**

create data that contain all kinds of sign language. Besides, I hope to achieve the real-time function of this system in the future.

Keywords – sigh language; translation system; Openpose; LSTM

I. INTRODUCTION

People who are lingual-disabled always have a feeling of loneliness, isolation, and frustration (World Health Organization, 2018) [1]. In most cases, they are unable to even access basic service and product (World Federation of the Deaf, 2016) [2]. Sign language is undoubtedly an effective tool for them to communicate with others but unfortunately, it is only understood by a few people.

There are multiple critical factors that determine the effectiveness and usefulness of sign language translation system. Although many kinds of research have been done to achieve goals such as chronological recognition, hand pose estimation, sentence translation, high accuracy, or real-time function by various methods, the popularization would be hindered if any of the factors misses. Pros and cons of several papers will be discussed and evaluated based on the five goals listed above.

In the paper of "Real-time American Sign Language Recognition with Convolutional Neural Networks" [3], the proposed method based on GoogLeNet achieves nearly 98% of accuracy with five letters and 74% with ten (Garcia & Viesca, 2016) [4]. To make the translator more accessible and faster, the researchers sacrifice the accuracy by using 2D color images. The translator could work on laptops in real-time, but it is not practical enough because there is only alphabet recognition. The same problem appears in another project-

Automatic Indian Sign Language Recognition for Continuous Video Sequence. However, given that this project exploits skin filtering which extracts the shape of hands, 24 different alphabets are recognized with the accuracy of 96% (Singha & Karen, 2015) [5]. CNN with skin segmented images could also be used for continuous sign language recognition (Tripathi & G.C.Nandi, 2015) [6]. Nevertheless, it's not considered as real continuous recognition. They used gradient based key frame extraction method and treated each frame independently, and then tried Euclidean Distance, Correlation and so on as classifiers. A weakness of this project is the high dependency on background and light condition.

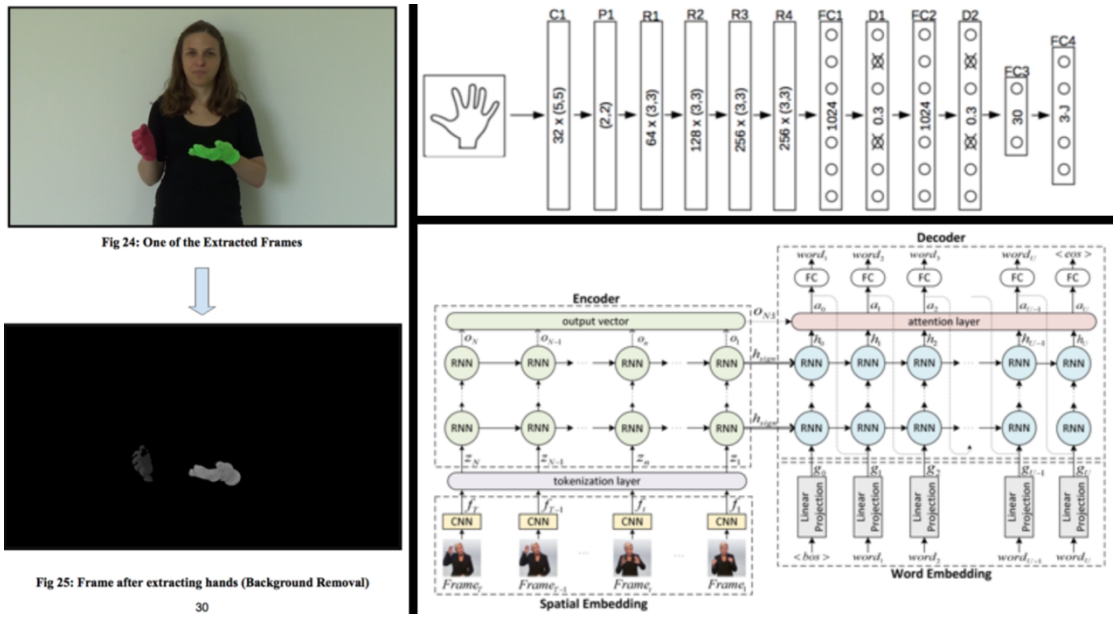


Fig. 1. Existing Projects. Left: special gloves used for hand pose estimation.(Cihan, 2018) [7]; Right Top: Resnet for hand pose estimation. (Masood, 2018) [8]; Right Bottom: A Sign Language Translation System. (Oberweger, 2017)

Recurrent neural network (RNN) could achieve the recognition of time-related action. In "Real Time Sign Language Gesture Recognition from Video Sequences" (Thuwal & Srivastava, 2017) [9], CNN is first used to get hand pose estimation from each frame of

spatial video. RNN is then used to gain the temporal features. It could finally achieve the accuracy of 95.217%. The advance of this project is that it employs CNN hand pose estimation by using depth image which significantly increase the accuracy hands recognition. However, it relies on depth videos with gloves which is not very accessible and may not apply well in daily life. Beyond of this project, there has already been a sign language translation product in the market. However, its tablets and depth camera are too expensive to afford or spread.

Researches give me a deep understanding of the feasibility and necessity of different methods. With pros and cons carefully considered, I decide to use the combination of CNN and RNN for sign language translation. Hand pose estimation could be achieved without depth camera by using OpenPose. Although hand pose estimation of OpenPose could increase accuracy, it will probably slow down the processing speed, so real-time will be a long-time goal. LSTM, an improved RNN network, will be used for chronological sign language recognition [10]. With OpenPose and LSTM network, the translation system could achieve sentence translation with 2D videos, as well as keep high accuracy, which would boost the popularization of sign language translation system.

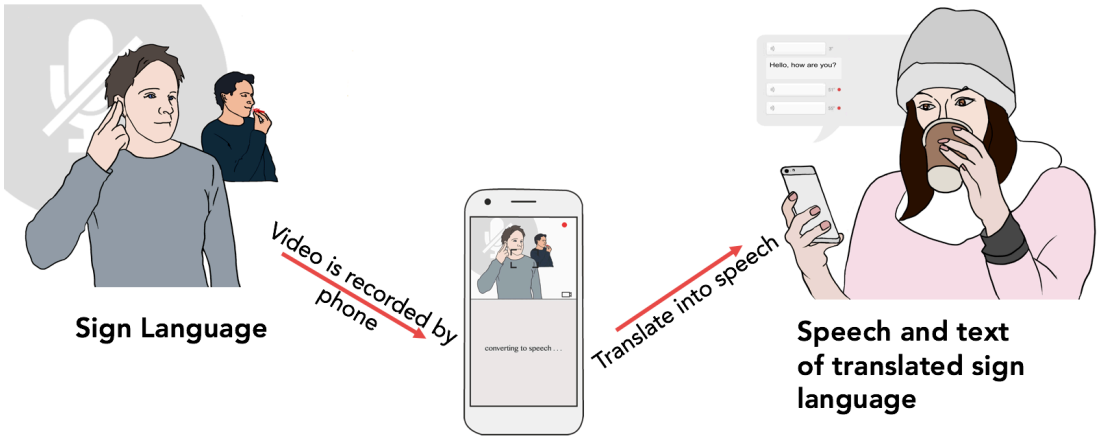


Fig. 2. Explanation of the Sign Language System

II. MATERIALS

In order to train and test the deep learning model, an American Sign Language dataset which primarily contains 340 videos completed by two subjects with 17 different phrases including both one-handed and two-handed signs was designed. The phrases are: Again, Deaf, Fine, Friend, Good, Hearing, Hello, I love you, I understand, My, Name, Nice to meet you, No, Nothing, Thank you, What's up, Yes. A dataset with 340 videos might not be enough to train a good model. So, the data are augmented through rotations, zooming, and translation (Fig. 3) whereby the number of video data is expanded to 3060. All the processed videos have the same resolution and fps as raw videos.

Fig. 3. Data Augmentation

III. METHODS

A. Recurrent Neural Network

Sign language translation is a kind of behavior recognition process which requires both feature extraction and feature recognition. The feature extraction process is successfully done by OpenPose, the convolutional neural network. During feature recognition process, a sign language video is complex sequences of extracted features of hands movement; in order to cope with the sequential information, RNN-LSTM model were introduced to include memory to model temporal dependencies.

RNN (the recurrent neural network) is an artificial neural network in which nodes in hidden layers are connected to the following hidden layers of a sequential data inputs. Input of a hidden layer contains not only input data but also output from previous hidden layer which processes data of previous frame of video, then would generate a new output. Consequently, the features of previous movements of hands are able to be considered in latter sign language recognition.

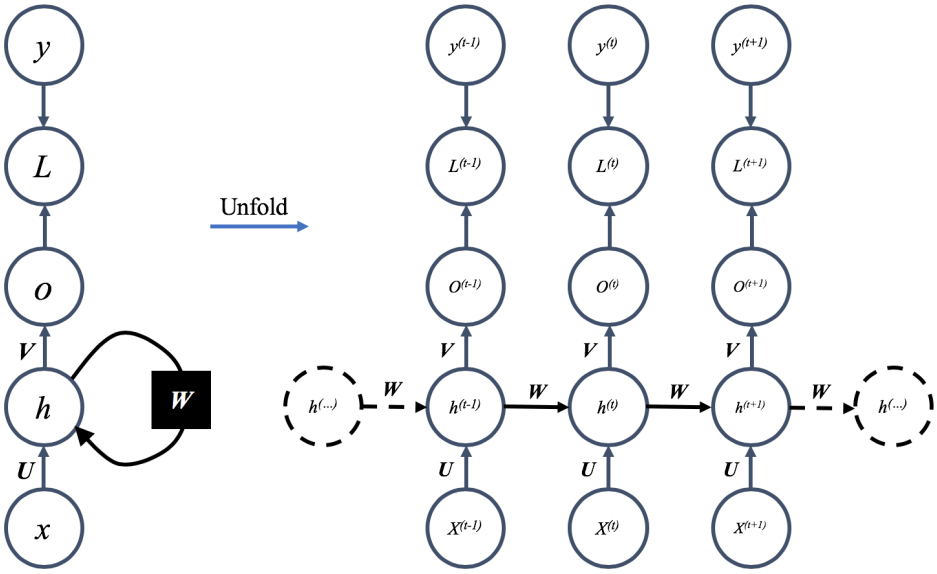


Figure 1: Representation of RNN Model

In an RNN model, $h^{(n)}$ is hidden-layer activations and $x^{(n)}$ is the inputs from sequential data. The input to hidden connection is parametrized by a weight matrix U, the hidden to hidden connection is parametrized by a weight matrix W, and the hidden to output connection is parametrized by a weight matrix V. The output $o^{(n)}$ is an unnormalized probability which is then put into $L^{(n)}$ to get $\hat{y} = \text{softmax}(o^{(n)})$. Finally, $\hat{y}$ is compared to $y^{(n)}$ to determine how far $o^{(n)}$ is from target.

B. Forward propagation:

The forward propagation equation for RNN can be expressed with a bias vector b:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$

Therefore, forward propagation has to begin with an initial state $h^{(0)}$. And the following $h^{(n)}$ can be calculated through activation function. Most of time, tanh function is used:

$$h^{(t)} = \tanh\left(a^{(t)}\right)$$

Then, the output is determined by $h^{(t)}$, weight matrix V, and bias vector c:

$$o^{(t)} = c + Vh^{(t)}$$

Finally, normalized probability is gained by:

$$\hat{y}^{(t)} = softmax(o^{(t)})$$

C. Loss Function:

The four equations can be applied on each time step from $t^{(1)}$ to $t^{(n)}$ so that the total loss of result of sequence x to the target sequence y can be determined by the sum of loses over all the time steps.

log loss function is used to evaluate how well the RNN models the dataset.

For each step of the RNN, the loss function of binary classification is:

$$L^{(t)}\left(\hat{y}^{(t)}, P(y^{(t)}|x)\right) = -log(y^{(t)})$$

By using logistic regression, $P(y^{(t)}|X)$ can be expressed as:

$$P(y^{(t)}|x) = \begin{cases} h_\theta(x) = g\big(f(x)\big) = \dfrac{1}{1+\exp\{-f(x)\}}\ ,\ \ y^{(t)} = 1 \\ 1 - h_\theta(x) = 1 - g\big(f(x)\big) = \dfrac{1}{1+\exp\{f(x)\}}\ ,\ \ y^{(t)} = 0 \end{cases}$$

Put this equation into previous one and we can get:

$$L^{(t)}\left(\hat{y}^{(t)}, P(y^{(t)}|x)\right) = \begin{cases} \log\left(1+\exp\{-f(x)\}\right)\ ,\ \ y^{(t)} = 1 \\ \log\left(1+\exp\{f(x)\}\right)\ ,\ \ y^{(t)} = 0 \end{cases}$$

The sum of loss function presents the overall loss function which is:

$$\mathrm{L}(\hat{y}, P(y|x)) = \sum_{t=1}^{T_{\hat{y}}} L^{(t)}\left(\hat{y}^{(t)}, P(y^{(t)}|x)\right)$$

In the case of sign language translation where the classification M is larger than 2, we calculate loss for each label and sum them together. The loss function for each step can be modeled as:

$$-\sum_{c=1}^{M} y_{o,c} \log\left(p_{o,c}\right)$$

M is the number of classes. Y is a binary indicator if observation *o* is correctly classified in the class c. p shows the probability that *o* is in class c.

D. Vanishing Gradient Problem of RNN

RNN has long-term dependencies. The problem happens during backward propagation where gradient of error from loss function is calculated in a backward order. By using chain rule of partial derivative, the gradient can be calculated as:

$$\frac{\partial L^{(t)}}{\partial U} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial \hat{y}^{(t)}} \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{j-1)}}\right) \frac{\partial h^{(k)}}{\partial U}$$

Because we are using tanh activation function $h^{(t)} = \tanh\left(a^{(t)}\right)$,

$$\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{j-1)}} = \prod_{j=k+1}^{t} \tanh' W$$

As the graph shows, the range of *tanh'* is between 0 and 1. Therefore, as t get bigger, product of the series of values that is between 0 and 1 will approach zero. Consequently, vanishing gradient would happen in long term.

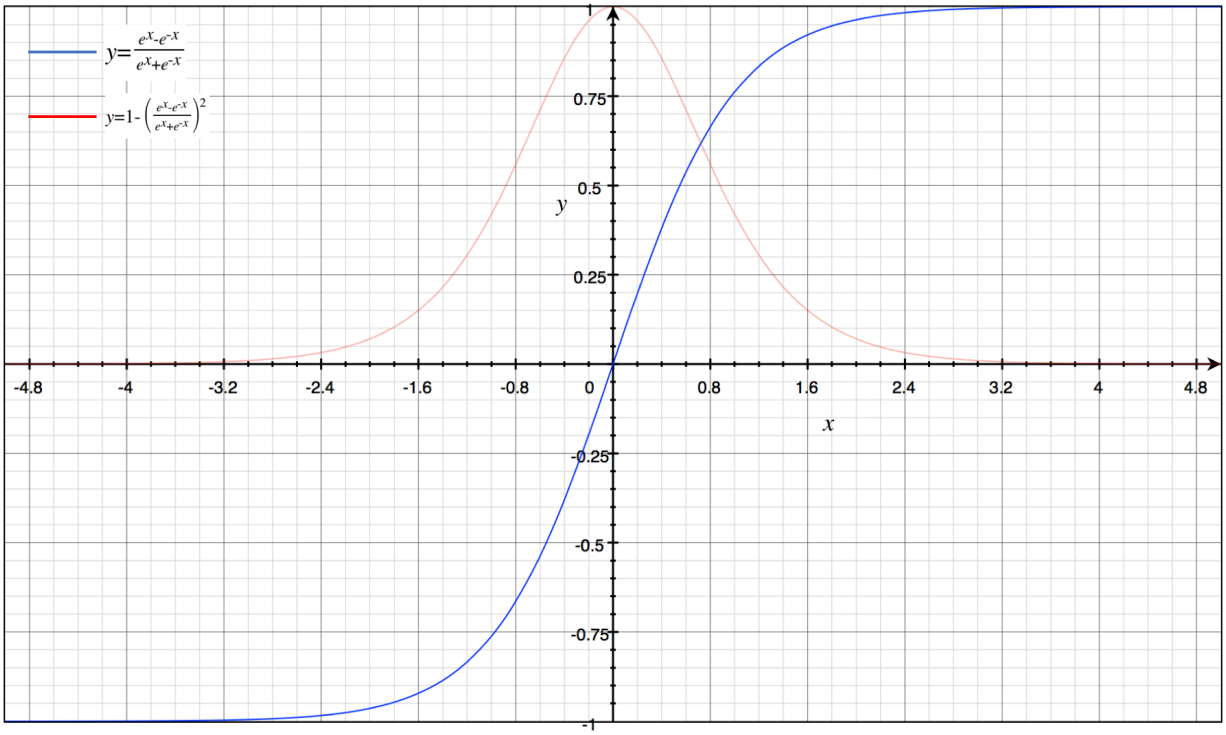
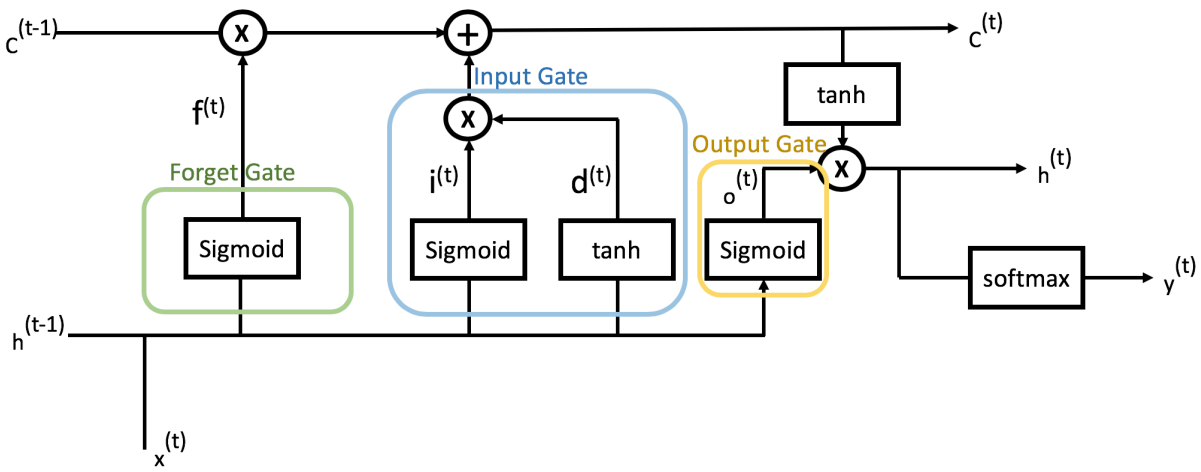


Figure 2: tanh function and its derivative

E. LSTM:



The structure of LSTM includes three gates. The forget gate determines what information from previous step should be kept or thrown. Output value from the sigmoid function is between 0 (less important) to 1 (very important). The input gate is used to control what information is important from

current input x. The output gate decides what the next hidden state should be. The math formula for $f^{(t)}$, $i^{(t)}$, $d^{(t)}$, $o^{(t)}$, $c^{(t)}$, $h^{(t)}$, and $y^{(t)}$ are:

$$f^{(t)} = sigm(W_{xf}x_t + W_{hf}h_{t-1})$$

$$i^{(t)} = sigm(W_{xi}x_t + W_{hi}h_{t-1})$$

$$d^{(t)} = sigm(W_{xd}x_t + W_{hd}h_{t-1})$$

$$o^{(t)} = sigm(W_{xo}x_t + W_{ho}h_{t-1})$$

$$c^{(t)} = f^{(t)} * c^{(t-1)} + i^{(t)} * d^{(t)}$$

$$h^{(t)} = o^{(t)} * \tanh(c^{(t)})$$

The forget gate creates a path for important information to be kept which solves the problem of vanishing gradient.

IV. DESIGN

A. Whole Design

The goal of this project is to get text and speech of translation result from an input video of sign language which separated into frames. Then, each frame is put into OpenPose for the spatial feature of hands. In order to extract the temporal feature from LSTM network, the spatial feature of each 30 continuous frames as an array is recorded. And every two successive arrays have 80 percent of overlaps. In the end, the LSTM network model will predict the most possible translation and speak it out. The overview of the system shows in Figure 4.

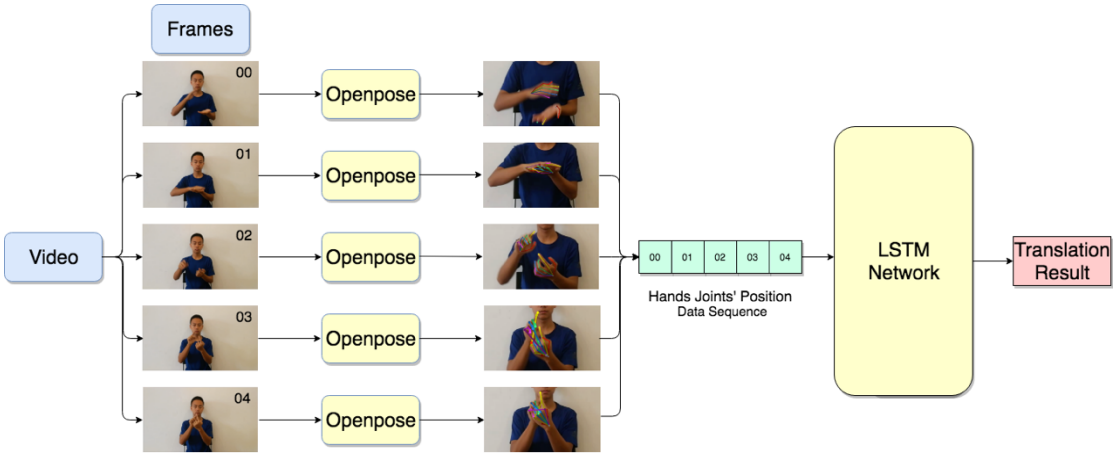


Fig. 4. Overview of the System

The development tools and environment as fellow:

Compiler: Jupyter Notebook 1.0.0

Environment: Python 3.6.5

Library: Tensorflow, OpenPose

GPU: GTX 1080

B. Spatial Feature Extraction

OpenPose designed by CMU is a real-time multi-person system to detect human key points on images. By using this library, a series of dictionaries containing x, y position and their predicted probability of 21 points on each hand could be generated. Extract the position information of each frame into a .txt document, which is the required spatial feature.

C. Temporal Feature Extraction

LSTM network has a great advantage in learning long-term memory tasks, so it could be effective in sequential data learning. In this neural network model, an RNN model based on two layers of LSTMs each of which contains 34 hidden layers is designed. The two layers are

followed with a layer of softmax activation, whose output is then matched with the expected training output.

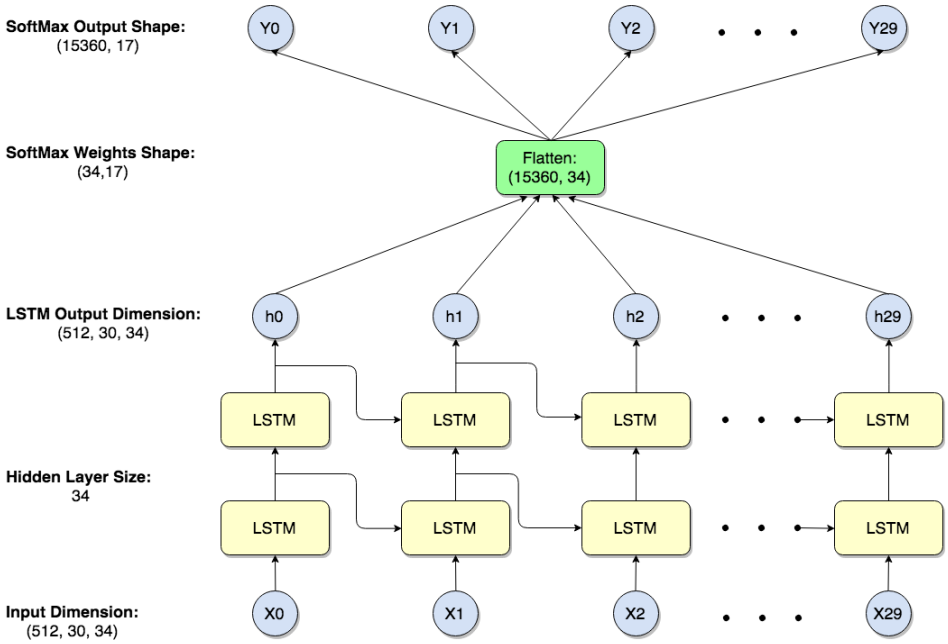


Fig. 5. LSTM Model

D. Model Improvement and Experiment Design

Regularization and dropout are used to reduce overfitting of the model. L2 regularization could increase the generalization ability of model and prevent overfitting by punishing high-value weights. Dropout is a technique which randomly selects and temporarily disables neurons. In this project, dropout of 0.8 and 0.5 is tried while other variables are controlled.

In order to find out better model, three variables, including hidden layers (34 layers and 45 layers), activation function (ReLU, Leaky ReLU, Elu and Tanh), and learning rate (0.5, 0.05, 0.005and 0.0005), are tested separately. Set epochs to 3000 and batch size to 512 for keeping the fairness of experiments. Only one independent variable will be changing each time.

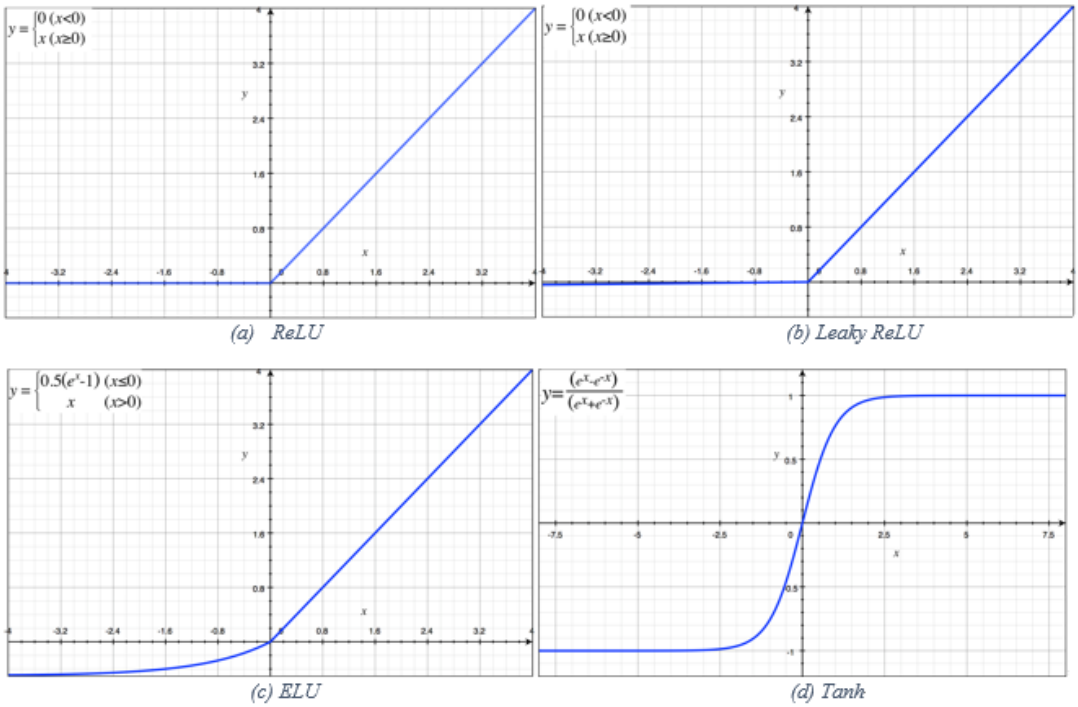


Fig. 6. Activation Functions.

V. Data Analysis

Trained by augmented data, the LSTM model has better performance on sign language recognition. From TABLE I, all metrics are approximately 10 percent higher after data augmentation. From the change value of precision and recall, it's obvious that augmented data helps the model to classify more true positives in relevant elements and less false positives in all positives.

TABLE I. Comparison of Accuracies

Dataset	Testing Accuracy	F1 Score	Recall	Precision
Before Augmentation (340 vidios)	81.80%	82.27%	81.80%	82.40%
After Augmentation (3060 vidios)	92.41%	92.44%	92.41%	92.61%

Based on the confusion matrix, before data augmentation, the decoding result of seventeen labels produced by the deep-learning annotator doesn't match with the ground truth. The classifier gets confused among easy signs ("deaf", "hearing", "I love you", "my", "no", "thank you", "yes"). On the opposite, it is good at recognizing complex signs ("nice to meet you" and "nothing"). It is caused by overfitting in which model loses the ability to generalize easy signs. When more data is used, the misclassification is largely eliminated.

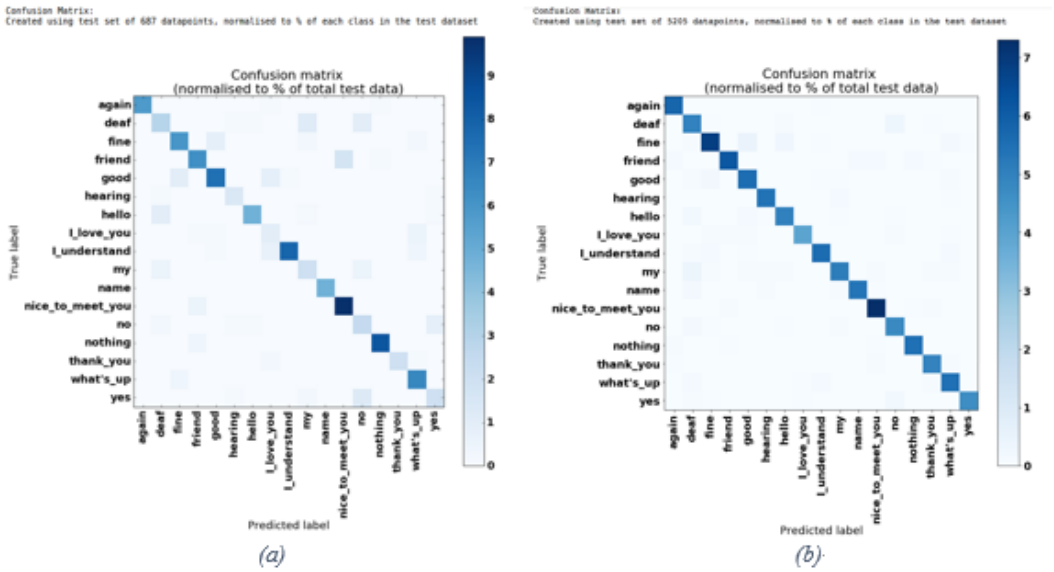


Fig. 7. Comparison of Confusion Matrix

Fig 8 contains curves of training accuracies and testing accuracies with the same model and different dataset. The training and testing accuracies have a gap in both situations. However, before data augmentation, the overfitting problem is very serious; the difference between training and testing accuracies is more than 15 percent. After data augmentation, the training and testing accuracies become closer. As a result, even though training accuracies is lower than the previous one, testing accuracy is 10.61% higher.

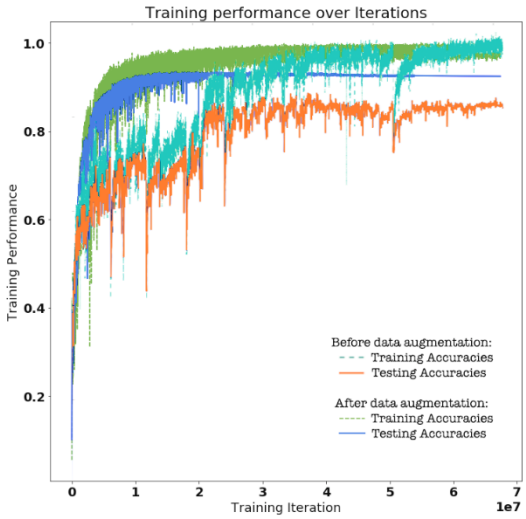


Fig. 8. Comparison of Training Performance

TABLE II. ACCURACIES OF THE PROPOSED SYSTEM

The Relationship Between Deep Learning Techniques and Validation Accuracy								
Version	**Activation Functions**	**Epoch**	**Hidden Layers**	**Learning Rate**	**Dropout**	**Batch Size**	**Validation Accuracy**	**F1 Score**
model 1.0	RELU	3000	34	0.005	N/A	512	86.32%	86.40%
model 1.0.1	RELU	3000	34	0.005	N/A	512	92.41%	92.44%
model 1.1	Leaky ReLU	3000	34	0.005	N/A	512	93.62%	93.66%
model 1.2	ELU	3000	34	0.005	N/A	512	92.07%	92.09%
model 1.3	Tanh	3000	34	0.005	N/A	512	85.94%	85.99%
model 2.0	Leaky ReLU	3000	34	0.5	N/A	512	91.34%	91.35%
model 2.1	Leaky ReLU	3000	34	0.05	N/A	512	88.78%	88.79%
model 2.2	Leaky ReLU	3000	34	0.0005	N/A	512	92.70%	92.73%
model 3.0	Leaky ReLU	3000	45	0.005	N/A	512	93.54%	93.58%
model 4.0	Leaky ReLU	3000	34	0.005	0.5	512	92.53%	92.55%
model 4.1	Leaky ReLU	3000	34	0.005	0.8	512	91.99%	92.03%

In order to improve the model, I tested activation functions, hidden layers, learning rate and learning rate decay, dropout, separately. The relationship of accuracy and each factor is revealed by these experiments. Because in sign language translation, both high accuracy and high recall is required, so the F1 Score is the best measurement of the effectiveness of models.

For activation functions, RELU, Leaky RELU, ELU and Tanh (Fig 6) are tested in this experiment. ReLU could prevent the vanishing gradient. However, many neurons might be permanently deactivated when parameters are smaller than zero. Leaky ReLU and ELU appear to solve this zero-value problem. In the result, the most effective activation function is Leaky RELU (model 1.1) which achieves 93.66% in F1 Score. And the lowest one is the Tanh because of the vanishing gradient. As for hidden layers, size of 34 and 45 is tested. The performance of the model with 34-layer size is pretty better because the deeper network would cause more serious vanishing gradient. In learning rates of 0.5, 0.05, 0.005, 0.0005, rate of 0.005 performs the best. Cause the neurons of this model are not small, dropout of 0.5 results in better performance.

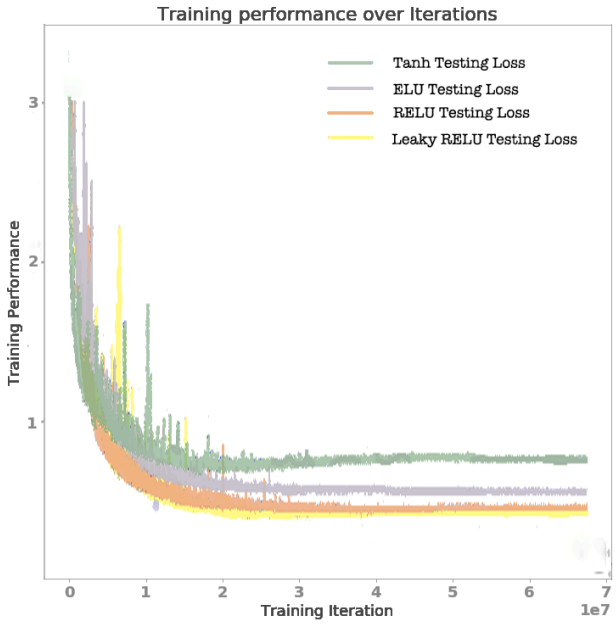


Fig. 9. Activation Function and Loss

Fig. 10. Different Learning Rate

In the result display, the video of a person who presents sign language and is not in the training dataset is put into the trained model through the interface built by Tkinter. The result of translation could be accurately predicted and speak out.

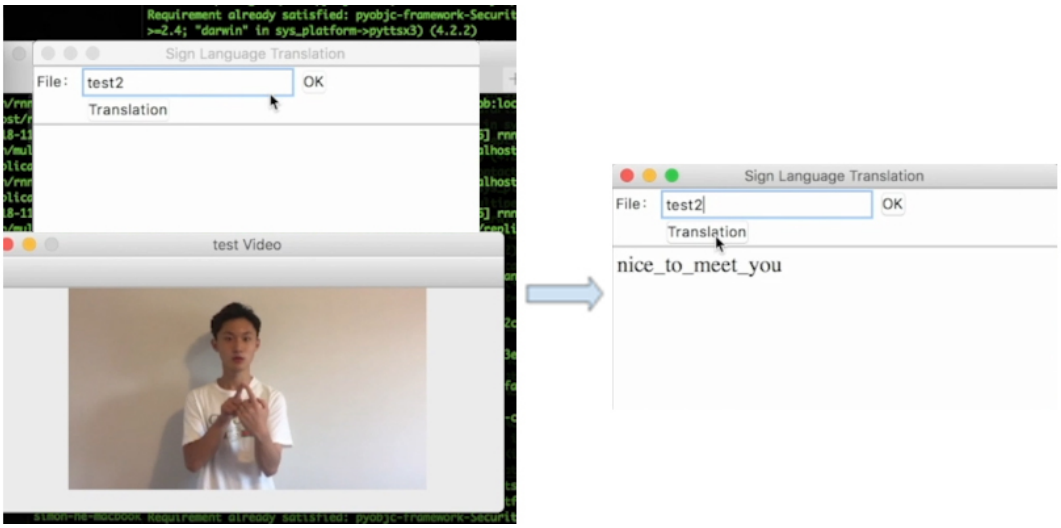


Fig. 11. The Display of Sign Language Translation System

VI. Discussion and Conclusion

In this project, I successfully developed a sign language translation system based on deep learning. This project could help deaf or mute to better communicate with others. The main parts of this project are:

- Create a sign language dataset containing 3060 videos of 17 different sign language phrases.
- Use Openpose for hand pose estimation.
- Build an LSTM network for sign language translation.

After the establishment and improvement of neural networks in 3 months, the f1 score of my translation system reaches 93.66%. The performance of this deep learning model is very high and achieves my previous goal of over 90%. My project is a successful solution for sign language translation:

- Supplement existing solutions of sign language translation system
  - Costly
  - Unreliable
  - Inaccessible
- Help sign language users to better communicate with others.
- However, this project also has some weakness:
- Real-time translation is not achieved right now.
- Dataset is not big enough.
- User interface is not well created.

These weaknesses could be solved in the future. In the next one year, I will work on the real-time translation and user interface. The further question is how to gain larger dataset of sign language.

REFERENCES

[1] World Health Organization. (2018, March 15). Deafness and hearing loss. Retrieved from World Health Organization: http://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[2] World Federation of the Deaf. (2016). Human Rights. Retrieved from World Federation of the Deaf: http://wfdeaf.org/our-work/human-rights-of-the-deaf/

[3] Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2016). Realtime multi-person 2d pose estimation using part affinity fields. arXiv preprint arXiv:1611.08050.

[4] Garcia, B., & Viesca, S. (2016). Real-time American sign language recognition with convolutional neural networks. Convolutional Neural Networks for Visual Recognition.

[5] Singha, Joyeeta, and Karen Das. "Automatic Indian Sign Language Recognition for Continuous Video Sequence." ADBU Journal of Engineering Technology 2, no. 1 (2015).

[6] Tripathi, K., & G.C.Nandi, N. B. (2015). Continuous Indian Sign Language Gesture Recognition and Sentence Formatio. Procedia Computer Science, Volume 54, 2015, Pages 523-531.

[7] Cihan Camgoz, N., Hadfield, S., Koller, O., Ney, H., & Bowden, R. (2018). Neural Sign Language Translation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7784-7793).

[8] Masood S., Srivastava A., Thuwal H.C., Ahmad M. (2018) Real-Time Sign Language Gesture (Word) Recognition from Video Sequences Using CNN and RNN. In: Bhateja V., Coello Coello C., Satapathy S., Pattnaik P. (eds) Intelligent Engineering Informatics. Advances in Intelligent Systems and Computing, vol 695. Springer, Singapore

[9] Thuwal, H. C., & Srivastava, A. (2017). REAL TIME SIGN LANGUAGE GESTURE RECOGNITION FROM VIDEO SEQUENCES. DEPARTMENT OF COMPUTER.

[10] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

BIBLIOGRAPHY

Bheda, V., & Radpour, D. (2017). Using deep convolutional networks for gesture recognition in American sign language. arXiv preprint arXiv:1710.06836.

Han, J., Award, G. M., Sutherland, A., & Wu, H. (2006, April). Automatic skin segmentation for gesture recognition combining region and support vector machine active learning. In 7th International Conference on Automatic Face and Gesture Recognition (FGR06) (pp. 237-242). IEEE.

Mekala, P., Gao, Y., Fan, J., & Davari, A. (2011, March). Real-time sign language recognition based on neural network architecture. In System Theory (SSST), 2011 IEEE 43rd Southeastern Symposium on (pp. 195-199). IEEE.

Oberweger, M., & Lepetit, V. (2017, August 28). DeepPrior++: Improving Fast and Accurate 3D Hand Pose Estimation. Markham, Ithaca, NY 14850, USA.

Ronchetti, F., Quiroga, F., Estrebou, C., & Lanzarini, L. (2016). Handshape recognition for Argentinian Sign Language using ProbSom. Journal of Computer Science & Technology; vol. 16, no. 1.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).