The 6th International Conference on Digital Arts, Media and Technology (DAMT) and
4th ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (NCON)

2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering | 978-1-6654-1569-9/20/$31.00 ©2021 IEEE | DOI: 10.1109/ECTIDAMTNCON51128.2021.9425711

Thai Sign Language Recognition: an Application of Deep Neural Network

Anusorn Chaikaew
Computer Science, SCIT, Chiang Rai Rajabhat University
Chiang Rai, Thailand
anusorn.c@crru.ac.th

Kritsana Somkuan
Information Technology, SCIT
Chiang Rai Rajabhat University
Chiang Rai, Thailand
kritsana@crru.ac.th

Thidalak Yuyen
Computer Science, SCIT Chiang Rai Rajabhat University
Chiang Rai, Thailand
thidaluk02@gmail.com

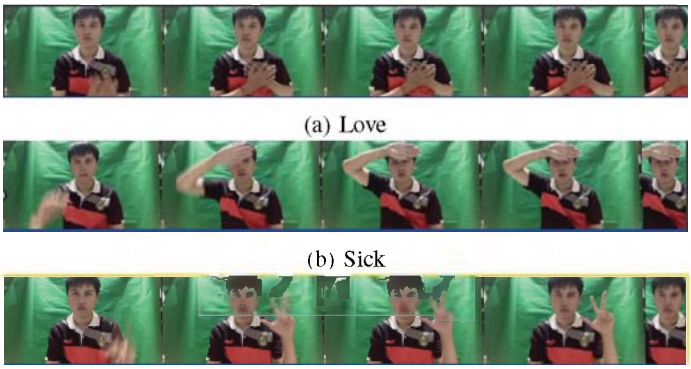
***Abstract*—Thai Sign Language (TSL) is the national sign language for Thai deaf people or hearing impaired in Thailand. These people with disabilities can use sign language to communicate with people with disabilities but face obstacles in communicating daily with ordinary people. Technology should play a key role in helping disadvantaged people achieve a better quality of life. This research aims to find ways to create Thai sign language recognition applications and be developed for real-time sign language translation in the next step. We purpose a simple approach with a MediaPipe framework that helps to extract the hand landmark from video on the preprocessing step and use that landmark to build the model for recognition hand gestures with various Recurrent neural networks (RNN). The result showed that the model builds with LSTM, BLSTM and GRU has an accuracy greater than 90 percent. This approach can produce an accurate close to the traditional approach.**

***Keywords*—Computer Vision , Machine Learning, Neural Network, Thai Sign Language (TSL)**

I. Introduction

A report on the situation of persons with disabilities in Thailand in September 2020 [1] found that there were 2,058,082 people or 3.09% of the entire population. Of these, 388,233 were hearing impaired or interpretive. Human-computer interaction (HCI) could help so many people at all levels of society to upgrade a better life, especially disabled people.

Sign Language is a subset of hand gestures that is part of the human modality to communication among deaf people or the hearing impaired. Thai Sign Language (TSL) was developed from American Sign Language (ASL) but different in gesture detail; for example, in Fig.1, an example of Thai sign language communicates the word love, sick and number three, respectively. TSL is a standard sign language in Thailand, but only some ordinary people understand it and use it to communicate with deaf people that maybe sometimes hearing impaired can not communicate in daily life with ordinary people. Sign Language Recognition (SLR) is the start point to develop sign language translation, which is interdisciplinary research area topics such as computer vision, natural language, artificial neural network processing, etc.



(a) Love

(b) Sick

(c) Three

Fig. 1. Example of a Thai Sign Language (TSL).

To allow the machine to recognize the sign language that uses hand movement, we need to prepare the dataset and design the training step with a suitable algorithm. Therefore, we decided to create our own data set to train the model, aiming to build the model that recognizes at least 100 gestures/word capability covering the vocabulary used in daily life in different categories such as Greetings, Numbers, Days, Times and Seasons, etc.

The most state-of-the-are algorithm recognizes body movement using a neural network to extract feature and trained model by decomposing the problem sub unit [2] or using cascade network [5]. This has led to our research aimed at using neural networks as well.

The organization of this paper is as follows. Section II contains some related work. We discuss some algorithm which is adopted or related in our work. Our approach is explained in Section III and evaluate the result in Section IV. Finally we conclude our contribution and discussing future work in Section V.

II. Related Work

One of the most used algorithms today and found to be effective for sign language recognition (SLR) is the deep neural network (DNN) implementation. In this paper, we adopted various algorithms under the DNN area to fit into our criteria. In particular, the RNN (Recurrent Neural Network) algorithm,

978-1-6654-1569-9/21/$31.00 ©2021 IEEE

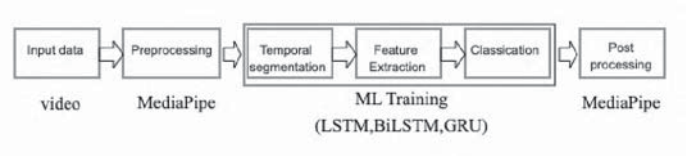


Fig. 2. Our approach.

since this research builds the modeling from extracted the key points on both palms when gesture movements in sign language.

A. Convolutional Neural Network

Convolutional neural networks (CNN), also called ConvNets, were first introduced in the 1980s by Yann LeCun is a class of deep neural networks most commonly applied to analyzing visual imagery or video. Sign language research area adopted CNN such as [2], [7], [8], [10] and so on for feature extraction and classification. Especially [10] focus on TSL like our paper showed accuracy up to 97.7 percent but based on the Kinect device for data acquisition step, unlike our research, the process of collecting the dataset does not require any special equipment.

B. Recurrent Neural Network

Recurrent Neural Network(RNN) is a class of artificial neural networks that mostly focus on temporal sequence data such as NLP, text, voice, video, or time-series data. So many research paper in machine translation using RNN or improved algorithm from RNN such as LSTM or GRU [4], and in this paper, we focus on using RNN since the extracted hand key point's position is saved in a text file format (CSV) corresponding to the reference position, it is more suitable for choosing RNN algorithms than CNN.

C. MediaPipe framework

MediaPipe [6] is a framework for building a pipeline mainly for rapid prototype base on C++. In this paper, we use MediapPipe for extracted hand key point that MediaPipe already provide by customize some source code and write the absolute position of hand landmark to text file base on guideline [11], [12].

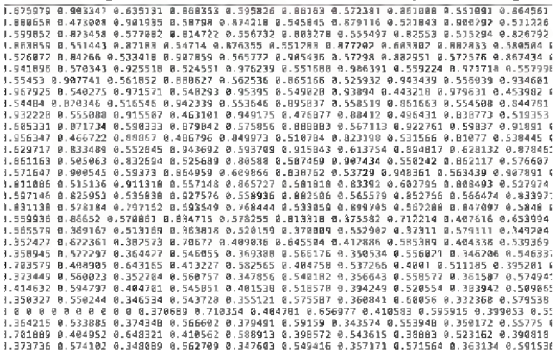
III. Our Approach

Sign Language mainly focus on two ways the feature extraction from images or extract hand key point and concatenated their location [9]. Our approach based on pipeline as Fig.2. It can be separated into four large steps: creating a video dataset. Preparing data to train the model; Model training and using models to create applications for predict Thai sign language; details are as follows.

A. Input Data

The very first step is to prepare a dataset to use for model training. We shoot 100 videos per word. However, in this experiment, we recorded five videos of five gestures, a total of

(a) 21 points on each hand detected by MediaPipe Framework



(b) After Extracted 42 key points and write to CSV files

Fig. 3. key point position detection on each palm and writing key point to text files (CSV)

500 videos, to test our approach. Each video contains 50 FPS with format H.264, and this process takes the longest time but does not rely on any additional equipment except a mobile camera.

B. Preprocessing

In this step, we customize source code from the MediaPipe framework to extracted the hand key point 3 (a) and write it to a text file (CSV) as Fig. 3 (b). At this stage, a total of 21 key locations on both palms, a total of 42 key points, will be recorded in a text file; all 42 landmarks will pull from each frame of the video. Therefore, we need each video to have the same lenght as possible to create an equivalent data set.

C. Model Training

At this stage, the hand's key point we extracted from the video is suitable for RNN, and we decide to use the three different variations of architecture related to RNN are LSTM (Long short-term memory), Bi-LSTM (Bidirectional LSTM), and GRU (Gated recurrent unit). LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Bi-LSTM is a compound of Long Short-Term Memory (LSTM) and Bidirectional Recurrent Networks (BiRNN). The GRU is like an LSTM with a forget gate but has fewer parameters than LSTM. Our approach in this step is that using a text file training set takes less time to train a model than the CNN approach that relies on the large image or video datasets, which can take a longer time than text file such as [10]. Moreover, this step also may be necessary to repeat the process many times by

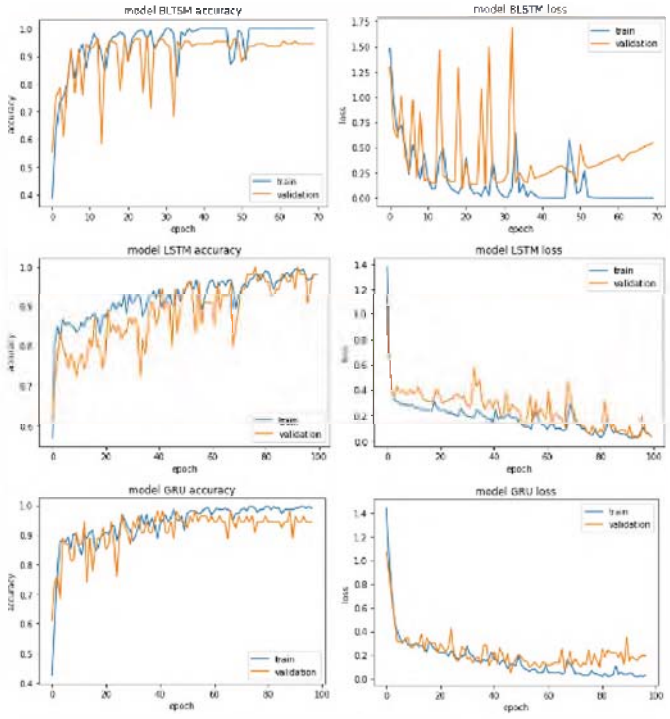


Fig. 4. Accuracy/Loss

TABLE I. Hyperband parameter for tuning each algorithm

parameter	min	max
input node	64	256
hidden layer	1	3
optimizer	'Adagrad','Adamax','Adam','RMSprop'	

adjusting the parameters to get the best accuracy. Thus text files might be easier to train model than other data set.

Additionally, In this step, we also are tuning the hyper-parameter with Hyperband [3] that focuses on speeding up random search through adaptive hyper-parameter allocation the importance Hyperband parameter list on Table I. From the table, we can see that we have the default number of nodes at 64 nodes and can be increased by a maximum of 256 nodes, and there are from 1 to 3 hidden layers and 4 types of optimizer for Hyperband to choose from. Fig. 5 show each RNN algorithm's summary after tuning with Hyperband. We found that the GRU had the least number of parameters while BiLSTM had the largest number of parameters, meaning BiLSTM was the largest model while the GRU was the smallest model in this experiment.

After tuning with Hyperband the next step is to build the model with training data and the output result of each algorithm after train shown in Table II. The data set of training data has 100 videos per word divided into 60 videos for training, 30 for the validation test, and the last 10 for testing the accuracy after trained models. All algorithms can surpass accuracy more than 90 percent with train data set and surpass more than 90 percent with test data set as Table III. During the model's practice with different epoch numbers ranging from

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 70, 128)           109056
_________________________________________________________________
lstm_1 (LSTM)                (None, 70, 192)           246528
_________________________________________________________________
lstm_2 (LSTM)                (None, 70, 64)            65792
_________________________________________________________________
dropout (Dropout)            (None, 70, 64)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 128)               98816
_________________________________________________________________
dense (Dense)                (None, 6)                 774
=================================================================
Total params: 520,966
Trainable params: 520,966
Non-trainable params: 0
```

(a) LSTM

```
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectional (None, 70, 256)           218112
_________________________________________________________________
bidirectional_1 (Bidirection (None, 70, 256)           394240
_________________________________________________________________
bidirectional_2 (Bidirection (None, 70, 256)           394240
_________________________________________________________________
bidirectional_3 (Bidirection (None, 128)               164352
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 1,171,718
Trainable params: 1,171,718
Non-trainable params: 0
```

(b) BiLSTM

```
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 70, 192)           160128
_________________________________________________________________
gru_1 (GRU)                  (None, 70, 192)           222336
_________________________________________________________________
dropout_1 (Dropout)          (None, 70, 192)           0
_________________________________________________________________
gru_2 (GRU)                  (None, 32)                21696
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
=================================================================
Total params: 404,358
Trainable params: 404,358
Non-trainable params: 0
```

(c) GRU

Fig. 5. Model Summary of each model after tuning with Hyperband

TABLE II. Model accuracy comparison while trained

Network	Accuracy	Loss
LSTM	0.94	0.16
BLSTM	0.97	0.39
GRU	0.93	0.20

70 to 100, we found that the graphs obtained while practicing Fig. Ref fig: acc-lose showed significant fluctuations in the BiLSTM, LSTM and slightly up and down on GRU due to the number of videos required for the practice (60 videos), it may not have been enough for BiLSTM or LSTM with parameters larger than GRU.

D. Post-processing

In this step, we developed both desktop and mobile applications, as shown in the Fig. 6 both of them have used different

TABLE III. Model Accuracy (Predicted with test data)

Model	Accuracy	Loss
LSTM	0.97	0.06
BLSTM	0.94	0.23
GRU	0.94	0.14

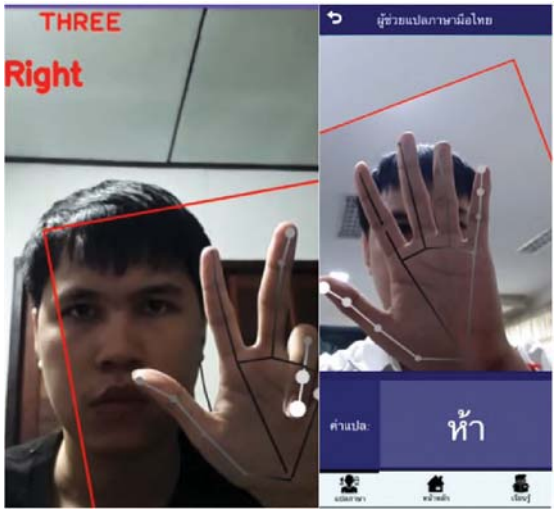


Fig. 6. Desktop Application (left) and Mobile Application (right)

models in the application development, choosing the model with the highest precision from the table. III, that is, LSTM has 97 percent accuracy for desktop. And the GRU model was chosen for mobile use because of its small size but acceptable accuracy.

IV. Result and Discussion

By looking at the results, we found that we could build and deploy the model in the development phase of both desktop and mobile applications. It shows that this approach can be applied to the development of both desktop and mobile sign language recognition. However, there are some limitations in binding to the MediaPipe Framework, but it may be acceptable in terms of open-source code.

V. Conclusion and Future Work

In this paper, we presented Thai Sign Languages recognition that builds from the MediaPipe framework by extracting key points from hand landmarks. We use just a simple smartphone camera and train it with CPU on the laptop and even faster when training with GPU. Our experiments demonstrate how to build Thai Sign Language recognition with the MediaPipe framework and will be the next step to make a Sign Language Translation and the next future.

Acknowledgment

This research has been supported by the Research and Development Institute of Chiang Rai Rajabhat University

References

[1] Department of Empowerment of Persons with Disabilities. (2020, October 24). A report on the situation of persons with disabilities in Thailand [Online]. Available: http://www.dep.go.th

[2] N. C. Camgoz, S. Hadfield, O. Koller, and R. Bowden, "SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition," in 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Oct. 2017, pp. 3075–3084, doi: 10.1109/ICCV.2017.332.

[3] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization," arXiv:1603.06560 [cs, stat], Jun. 2018, Accessed: Nov. 02, 2020. [Online]. Available: http://arxiv.org/abs/1603.06560.

[4] K. Cho et al., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," arXiv:1406.1078 [cs, stat], Sep. 2014, Accessed: Nov. 02, 2020. [Online]. Available: http://arxiv.org/abs/1406.1078.

[5] D. Chen, G. Hua, F. Wen, and J. Sun, "Supervised Transformer Network for Efficient Face Detection," arXiv:1607.05477 [cs], Jul. 2016, Accessed: Nov. 02, 2020. [Online]. Available: http://arxiv.org/abs/1607.05477.

[6] C. Lugaresi et al., "MediaPipe: A Framework for Building Perception Pipelines," arXiv:1906.08172 [cs], Jun. 2019, Accessed: Nov. 02, 2020. [Online]. Available: http://arxiv.org/abs/1906.08172.

[7] P. Kishore, G. A. Rao, E. K. Kumar, M. T. K. Kumar, and D. A. Kumar. Selfie sign language recognition with convolutional neural networks. International Journal of Intelligent Systems and Applications, 10(10):63, 2018.

[8] H. Shin, W. J. Kim, and K.-a. Jang. Korean sign language recognition based on image and convolution neural network. In Proceedings of the 2nd International Conference on Image and Graphics Processing, pages 52–55. ACM, 2019.

[9] D. Li, C. R. Opazo, X. Yu, and H. Li, "Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison," arXiv:1910.11006 [cs], Jan. 2020, Accessed: Oct. 25, 2020. [Online]. Available: http://arxiv.org/abs/1910.11006.

[10] N. Sripairojthikoon and J. Harnsomburana, "Thai Sign Language Recognition Using 3D Convolutional Neural Networks," in Proceedings of the 2019 7th International Conference on Computer and Communications Management, Bangkok Thailand, Jul. 2019, pp. 186–189, doi: 10.1145/3348445.3348452.

[11] M. Tachionstrahl, "Tachionstrahl/SignLanguageRecognition," 27-Oct-2020. [Online]. Available: https://github.com/Tachionstrahl/Sign LanguageRecognition. [Accessed: 11-Nov-2020].

[12] A. Kim, "rabBit64/Sign-language-recognition-with-RNN-and-Mediapipe," Apr-2020. [Online]. Available: https://github.com/rabBit64/Sign-language-recognition-with-RNN-and-Mediapipe. [Accessed: 01-Oct-2020].