The Performance of Thai Sign Language Recognition with 2D Convolutional Neural Network Based on NVIDIA Jetson Nano Developer Kit

Eakbodin Gedkhaw

Faculty of Management Science, Chandrakasem Rajabhat University, Thailand

Abstract **– Thai Sign Language Recognition is a Thai Sign Language learning computer recognition. The system constructs an architecture of T-SLR by TSR-2DCNN based on NVIDIA Jetson Nano Developer Kit. It is a novelty of automatic translation TSL innovation and reveals the performance of feature extraction and classification to reduce crashed system, overloaded or automatic reboot while complicated processing occurs. The dataset contains 7 gestures in TSL, training images are 7,000 images and validation images are 700 images. The result compares with many techniques as shown that TSR-2DCNN can increase the performance of T-SLR in real-time, effectiveness with an accuracy of 0.9914 and loss of 0.03537.**

Keywords **– Thai Sign Language, Convolutional Neural Network, NVDIA Jetson Nano, Gesture Image Segmentation, Classification.**

1. Introduction

Thai Sign Language (TSL) is a visual natural language for connection or sharing information among people with hearing impairment [1], [2].

DOI: 10.18421/TEM111-52
https://doi.org/10.18421/TEM111-52

Corresponding author: Eakbodin Gedkhaw,
Faculty of Management Science, Chandrakasem Rajabhat University, Thailand
Email: s5907011910070@email.kmutnb.ac.th

Received: 24 November 2021.
Revised: 13 February 2022.
Accepted: 18 February 2022.
Published: 28 February 2022.

It uses hand gesture and shows meaning of word or alphabet. In Thailand, TSL is an efficient communication way for disabled people in learning. The analysis of human movement, human-computer interaction (HCI) and user's interface [3] can enhance the performance of Thai Sign Language Recognition (T-SLR). Moreover, computer can interpret into text [4] which rise interaction ability between computer and human.

At the present time, T-SRL uses image processing for training approach. The advantage is that complicated equipment is unnecessary, but the pre-processing and sensor device [5], for example, Kinect Sensor [6], Leap Motion [7], [24], Data Glove [8], [9] and Surface EMG [10], [11] have considerable importance. The raw data from sensor device input directly into this phase and then process start to learn TSL gesture. For TSL, it has to applied gestures to enable groups of deaf people in Thailand to communicate with each other. In 2020, Thailand was found that 391,785 men from the total 2,076,313 or 18.87% are hearing impairments. It challenges to discover automatic T-SRL [21] to support communicating with the hearing impaired and enabling the hearing impaired to interact with computers.

As for AI development toolkit and machine learning by NVIDIA Jetson Nano Developer Kit [12] or Kit-A02, there is a small powerful processor on GPU from NVIDIA corp. which can run numerous neural networks at the same time and use only 5 watts of power [13]. Moreover, Kit-A02 also supports AI/machine learning frameworks. It designs for new technology learners to implement AI applications on small board easily. This board can simply plug in robot or AI devices to increase capabilities, such as object recognition [14], gesture recognition, speech processing and automatic driving. This research base on Kit-A02 is created to extract feature of TSL by 2D-Convolutional Neural Networks (TSR-2DCNN). The researcher found that there is many research in TSL and its application

such as skin detection [25], principal component analysis (PCA) [15], one-shot learning hand gesture recognition [16], artificial neural network (ANN) [26], multilayer perceptron (MLP) [22], hidden makov model (HMV) [23] and the most popular is convolutional neural network [17].

The researcher proposes TSR-2DCNN architecture in this paper and up-down layer sizes in convolutional neural network suitable for processing base on Kit-A02. The board major problem is when the complicated processing and large scale started, it was crashing, overloading, and rebooting immediately. Therefore, the researcher introduces new TSR-2DCNN to suitably solve this problem and computational intelligence that can recognize TSL correctly and speedy. We can see that the feature extraction increases in real-time. The next part is application of NVIDIA Jetson Nano Developer Kit, convolutional neural networks, and gesture image segmentation. Then, proposed methodology shows methodology in research, illustrates performance of TSR-2DCNN in feature extraction of TSL. Then, research results include performance of TSR-2DCNN. And finally, we will present discussion and conclusion in future work.

2. The Application with Jetson Nano Developer Kit

Jetson Nano Developer Kit or Kit-A02 was developed by Nvidia for artificial intelligence and machine learning development. Kit-A02 is designed for robotics technology suitable for installation on smaller platforms. The board measures 70 x 45 mm, based on a single-chip Erista system with 4 Cortex-A57 cores, an operating frequency of 1.43 GHz, and a graphics system with 128 CUDA cores on Maxwell architecture [18]. The compute power of the GPU is 472 Gflops, has 4 GB of LPDDR4 RAM and a 16 GB eMMC drive. It also has USB 3.0, DisplayPort and HDMI video output, RJ-45 network port, slot M.2, supports SDIO, UART, I2C, GPIO, SPI and MIPI-CSI. Figure 1 shows architecture of Kit-A02 and Figure 2 shows the operation of Kit-A02.

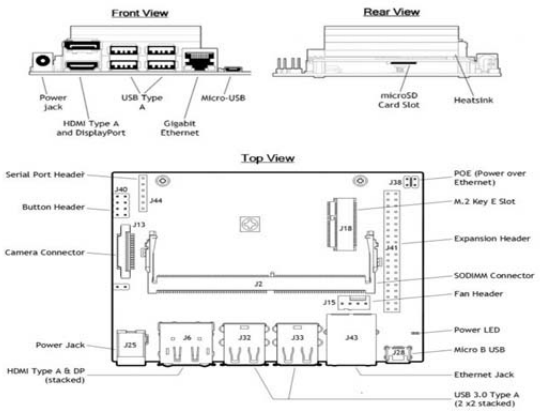


Figure 1. Kit-A02 architecture [19]

Kit-A02 is designed for the new technology learners and can get started as soon as a microSD card with System Image is inserted since a computer board is used by robots or other AI devices. The device will increase object recognition, image classification, object segmentation, image or speech processing and automatic driving abilities when merge supported devices without connecting to cloud system. It uses 5 watts of power and can also be powered by Micro-USB to connect to wide variety sensors that can run AI frameworks and application software from the NVIDIA JetPack SDK. It can run multiple neural networks at the same time.

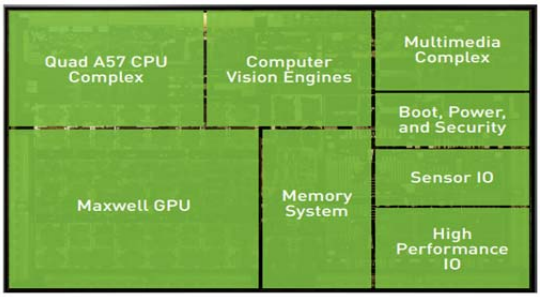


Figure 2. ervice of Kit-A02 [29]

In this research, Figure 3 shows performance of T-SLR by using TSR-2DCNN based on Kit-A02 to extract feature and classification in T-SLR.

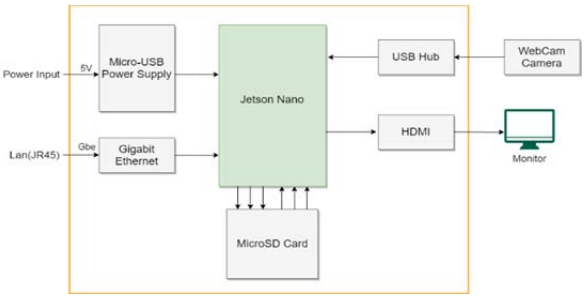


Figure 3. The application of Kit-A02 board

1. J501 Slot using for microSD card.
2. Ethernet uses J43 as local area networks (LANs) and the speeds that network can support 10Mbps/100Mbps/1000Mbps.
3. J6 is HDMI port for interface with display the extent to which two independent systems converge and operate or communicate with each other.
4. USB 3.0 Type A uses J32 and J33, which are dual-mode interfaces, Plug-and-Play, between computer and accessories, including WebCam, keyboard and mouse.
5. Micro-USB Power Supply provides 5.25V power which is the key point of Jetson Nano board requires a minimum 4.75V power to operate. Power supply can supply 5V at J28 Micro-USB connector.

3. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is artificial neural networks used to extract feature and classify high information dimensional image. The main feature is extract feature and classify information directly. It reasonable for image and VDO recognition [27]. CNN was developed from multilayer perceptron [28] and applied by down-size data dimension to get rid of unnecessary parameters and down-scale artificial neural network. Additionally, CNN reduces total training time which uses less resource in processing. Back propagation is trendy for recognition. Furthermore, the distinctiveness was automatically extracted by masked convolutional neural network generated from weight of feature maps. The architecture of CNN is shown in Figure 4.

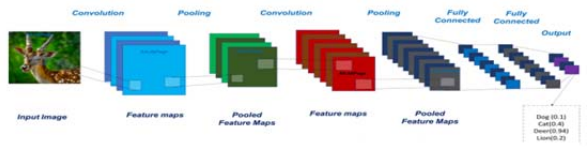


Figure 4. The architecture of CNN [20]

There are 4 layers in CNN as follows:

3.1. Convolution Layer. is the first layer in CNN. Its purpose is to extract object feature by convolution value of each kernel which obtain from training. Each kernel is distinctly feature extraction. The first convolution layer obtained low level feature such as edges, lines, and angles etc. The next convolution layers will receive high level features such as multiple line merge.

3.2. Rectified Linear Units layer (ReLU layer). This layer has an activation function to decide which node is the next to send data as equation follows,

$$(x) = \max(o, x)$$

3.3. Pooling layer. Subsampling data was occurred in this layer to decrease data variance and node in hidden layer. It calculates the maximum or average of each not overlapped area. The output is large scale feature which will down-size because it fits for detection and recognition

3.4. Dropout layer. This layer is the hardest training layer because it has to dwindle down the repetitive remembering model by sampling probability value between 0.5-1.0 in each neuron. The disconnected neuron does not transfer data into next layer and can increase speed of training also.

CNN includes convolution layers that describe extract feature. It starts with convolution layer and is followed by pooling layer, and ends with one or two fully connected layer or SoftMax. One neuron in SoftMax layer is complicated and dense because previous input from layer was multiply by connected weight and bias. The innermost part of the neuron is a non-linear activation function. Training algorithm in CNN is backpropagation by which n layer has label as [L_1, L_2, …, L_n], the output layer lth in L_l adds neurons. Considering neuron i^{th} which activated A_i^l, output neuron j in layer $(l + 1)^{th}$ follow by equation

$$y_j^{(l+1)} = \sum_i W_{ij}^l A_i^l + b_j^l$$

where W_{ij}^l is adjust weight connection between neuron i^{th} and neuron j^{th} in layer l and $(l + 1)^{th}$ b_i^l is bias value of neuron j^{th}, $A_j^{(l+1)} = f(y_j^{(l+1)})$ is activation function which f (·) is activation function for vector training feature and labelled set {(x(1), l(1)), (x(2), l(2)), ... , (x(m), l(m))}

The gradient output $y_j^{(n)}$ calculate as equation (3)

$$g_j^{(n)} = \frac{\partial x}{\partial A_j^{(n)}} C\left(W, b; x_{(k)}, l_{(k)}\right) f'\left(y_j^{(n)}\right)$$

The gradient $g_j^{(n)}$ can expand back to output layer L_n to layer L_2 for neuron j^{th} and layer L_l will be assign as equation (4)

$$g_j^{(l)} = \left(\sum_{h=1}^{k} W_{jh}^{(l)} g_h^{(l+1)}\right) f'\left(y_j^{(n)}\right)$$

The proper weight W will calculate in gradient and bias value will calculate by partial derivative.

$$\frac{\partial C}{\partial W_{ij}^{(l)}} = A_i^{(l)} g_j^{(l+1)}$$

$$\frac{\partial C}{\partial b_j^{(l)}} = g_j^{(l+1)}$$

Each repetitive epoch, W and b will be improved by negative value of average gradient from equation (5) and (6) in higher-level classification convolution layer by replacing data of each pixel with neighbor pixel.

This study applied 2D-CNN by the activation value at the spatial position (x,y) in j mapped characteristic of layer i represented by $v_{i,j}^{x,y}$ is generated using equation (7)

$$v_{i,j}^{x,y} = \emptyset\left(b_{i,j} + \sum_{\tau=1}^{d_{l-1}} \sum_{\rho=-\gamma}^{\gamma} \sum_{\sigma=-\delta}^{\delta} w_{i,j,\tau}^{\sigma,\rho} \times v_{i-1,\tau}^{x+\sigma,y+\rho}\right)$$

where φ is the activation function; $b_{i,j}$ is a bias parameter for j which mapping feature layer i, $d_{l-1}$is the number of mapping feature layer (l-1) and the depth of Kernel $w_{i,j}$, for j in mapping feature layer l; 2γ+1 is the kernel width, 2δ+1 is the kernel height and $w_{i,j}$ is weight of parameter j in mapping feature layer i.

4. Gesture Image Segmentation

Gesture image segmentation was used in this experiment. This step goes on after cropped image TSL gesture from webcam. Recognition phase will train particularly area to detect hand gesture only. Background and environment are important to be avoidable. These can use gesture image segmentation [25] which is hand gesture segmentation technique [26] by fragment distinct gesture frame. To eliminate background, one frame has to be cut off from other frames and every large difference label to detect area. It can detect edges of object movement; in this case it is body movement. However, pixel of density light has been changed if light condition has rapidly changed. Moreover, noise will rise when it is output. Figure 5 shows hand gesture image segmentation.

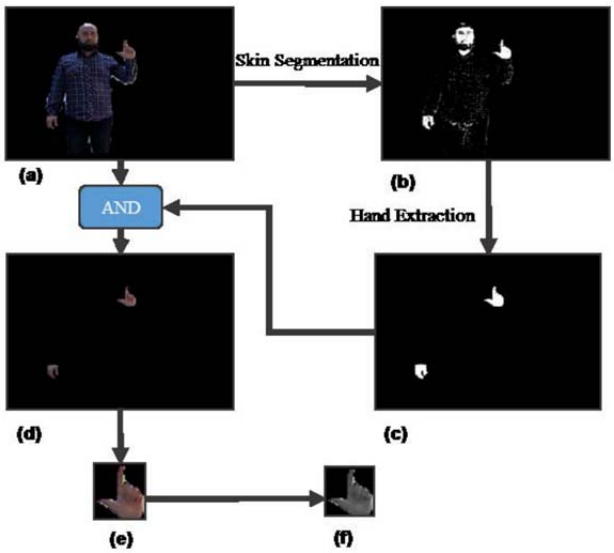


Figure 5. Gesture image segmentation [27]

5. Proposed Methodology

This section describes how to apply TSR-2DCNN based on Kit-A02 in experiment to extract TSL feature. It consists of 4 steps start with pre-processing. This step is received RGB image from Webcam and cleaning image to be sharpen. Second step is gesture segmentation only within concentration area. The third step is feature extraction, to extract hand gestures feature for communication in training system. The last is classification and forecasting. Figure 6 shows steps in experiment and Figure 7 shows toolkit.

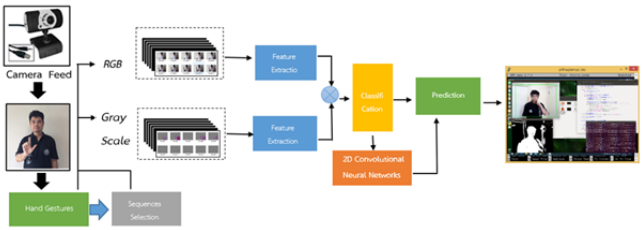


Figure 6. The step in TSR-2DCNN

Figure 7. Toolkit is TSL-2DCNN

5.1. Proposed TSR-2DCNN

In generally, CNN has many layers, and processing consists of convolution, ReLU, Max Pooling and fully connected layer. In the research, the researcher applied 2D-CNN in feature extraction TSL. The structure of TSL-2DCNN is shown in Figure 8.

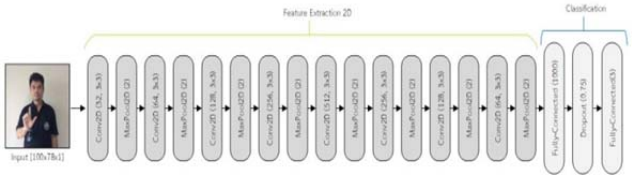


Figure 8. TSL-2DCNN architecture

Table 1 shows TSR-2DCNN architecture to evaluate the T-SLR performance using 2D Convolutional Neural Network Base on Nvidia Jetson Nano Developer Kit.

Table 1. TSR-2DCNN architecture

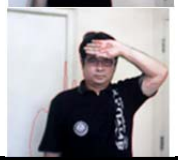
Layers	Layer Configuration	Output Size
Input	-	78, 100, 1
Conv2D	32 filters, 3 x 3 kernal and ReLU	78 x 100 x 32
MaxPool2D	32 filters, 2 x 2 kernal and ReLU	39 x 50 x 32
Conv2D_1	64 filters, 3 x 3 kernal and ReLU	39 x 50 x 64
MaxPool2D_1	64 filters, 2 x 2 kernal and ReLU	20 x 25 x 64
Conv2D_2	128 filters, 3 x 3 kernal and ReLU	20 x 25 x 128
MaxPool2D_2	128 filters, 2 x 2 kernal and ReLU	10 x 13 x 128
Conv2D_3	256 filters, 3 x 3 kernal and ReLU	10 x 13 x 256
MaxPool2D_3	256 filters, 2 x 2 kernal and ReLU	5 x 7 x 256
Conv2D_4	512 filters, 3 x 3 kernal and ReLU	5 x 7 x 512
MaxPool2D_4	512 filters, 2 x 2 kernal and ReLU	3 x 4 x 512
Conv2D_5	256 filters, 3 x 3 kernal and ReLU	3 x 4 x 256
MaxPool2D_5	256 filters, 2 x 2 kernal and ReLU	2 x 2 x 256
Conv2D_6	128 filters, 3 x 3 kernal and ReLU	2 x 2 x 128
MaxPool2D_6	128 filters, 2 x 2 kernal and ReLU	1 x 1 x 128
Conv2D_7	64 filters, 3 x 3 kernal and ReLU	1 x 1 x 64
MaxPool2D_7	64 filters, 2 x 2 kernal and ReLU	1 x 1 x 64
Fully Connected	-	1000
Fully Connected	-	3

Table 1 describes detail of TSR-2DCNN, which begins with input data use shape [None, 78, 10, 1] and pass through first layer, convolutional layer, as known Conv2D layer. This layer contains 32 feature maps and kernel size of 3x3. MaxPooling2D is called MaxPool2D and contains 32 feature maps with a size of 2×2. Convolutional layer called Conv2D_1 layer contains 64 feature maps with a size of 3×3. MaxPooling2D layer called MaxPool2D_1 contains 64 feature maps with a size of 2×2. Convolutional layer called Conv2D_2 layer, contains 128 feature maps and its size is 3×3. MaxPooling2D layer called MaxPool2D_2, contains 128 feature maps with a size of 2×2. Convolutional layer called Conv2D_3 layer, contains 256 feature maps with a size of 3×3. MaxPooling2D layer called MaxPool2D_3 layer, contains 256 feature maps with a size of 2×2. Convolutional layer called Conv2D_4 layer, contains 512 feature maps with a size of 3×3. MaxPooling2D layer called MaxPool2D_4 layer, contains 512 feature maps and its size is 2×2. Convolutional layer called Conv2D_5 layer, contains 256 feature maps with a size of 3×3. MaxPooling2D layer called MaxPool2D_5 layer, contains 256 feature maps with a size of 2×2. Convolutional layer called Conv2D_6 layer, contains 128 feature maps with a size of 3×3. MaxPooling2D layer called MaxPool2D_6 layer, contains 128 feature maps with a size of 2×2. Convolutional layer called Conv2D_7 layer, contains 64 feature maps with a size of 3×3. MaxPooling2D layer called MaxPool2D_7 layer, contains 64 feature maps with a size of 2×2. The above layers are activated with ReLU. The next layer is fully connected layer called Fully Connected layer, which contains 1000 neurons. And the last is fully connected layer called Fully Connected layer, contains 3 neurons.

5.2. Dataset

This section describes how to prepare data set in experiment. TSL gestures uses in this experiment was produced by the researcher. 7 gestures of TSL were chosen, “Hello”, “Day”, “Salted”, “Deer”, “Love”, “Stomach” and “Sick”. Total images in training phase are 7000 images, and 700 images in validation phase. The image size is 100*78 from webcam is converted into grey scale. Filtering is done by threshold and segment by labelled into binary code. Table 2 describe TSL data set in experiment.

Table 2. Dataset for TSR-2DCNN

Sign Language	Feature Extraction	Meaning	Labels
		Hello	[1,0,0]
		Day	[0,1,1]
		Salted	[1,1,0]
		Sick	[0,0,1]
		Deer	[1,1,1]
		Love	[0,1,0]
		Stomach	[1,0,1]

5.3. Training and Testing

This paper explores the performance of continuous T-SLR using TSR-2DCNN. The logarithm loss function and ADAM gradient descent were used to train in experiment. Total images in training phase are 7000 images and validation phase are 700 images with a size of 100*78 from webcam. The learning rate was initially set to 0.001, evaluate model by multilayer perceptron. This CNN is fit for 50 epoch and updates every 100 images. Next step is to assess the accuracy performance in T-SLR model with data set to extract features of TSL correctly. Next step comprises analyses' result and evaluation of the performance of the model and conclusion in the final. All of procedures are processed on NVIDIA Maxwell GPU, Quad-core ARM Cortex-A57 MPCore processor and 4 GB 64-bit LPDDR4 Memory. The accuracy value use to measure performance of TSL classification by percentage. Conclusion, discussion and comparison include Naïve Bayes, Decision Tree, KNN, Logistic Regression, Random Forest, SVM, ResNet-20 and ResNet-30 to evaluate the performance of TSR-2DCNN in continuous TSL.

6. Experiments & Result

The experiment used Kit-A00 on GPU with 128-core NVIDIA Maxwell, memory which is LPDDR4 4 GB, 64-bit 25.6 GB/s and 16 GB eMMC drive. The OS is Ubuntu 18.04 LTS aarch64 in JetPack 4.2.1 with L4T R32.2 (K4.9) and CUDA 10.0.326. Python is implementation language for evaluating the effectiveness of TSR-2DCNN in continuous independent T-SLR. Additionally, OpenCV and gesture segmentation were used in experiment. The results of T-SLR using TSR-2DCNN has an accuracy of 0.9914 and loss of 0.03537, and training time is 2 hrs. 04 min 54 sec and as shown in Figure 9.

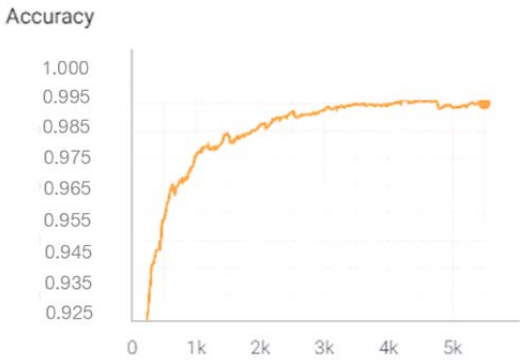


Figure 9. The accuracy of T-SLR.

Figure 9 showed accuracy value of T-SLR using TSR-2DCNN. It shows that the system has more accuracy when epoch has continuously increased. Thus, model is suitable for T-SLR when process is in real-time efficiency and shown confusion matrix result is in Table 3.

Table 3. Confusion Matrix of the proposed TSR-2DCNN model

	Hello	Day	Salted	Deer	Love	Stomach	Sick	None
Hello	100	0	0	0	0	0	0	0
Day	0	100	0	0	0	0	0	0
Salted	0	0	99	0	0	0	0	1
Deer	0	0	0	98	2	0	0	0
Love	0	0	0	1	99	0	0	0
Stomach	0	0	0	0	0	98	0	2
Sick	0	0	0	0	0	0	100	0
None	0	0	0	0	0	0	0	0

Table 3 showed the segmentation result of TSL using TSR-2DCNN, gesture words such as “Hello”, “Day” and “Sick”. It obtains accuracy of 1.0, “Salted” and “Love” obtains accuracy of 0.99. “Deer” and “Stomach” obtain accuracy of 0.98 respectively. Despite, when TSR-2DCNN is compared with Naïve Bayes, Decision Tree, KNN, Logistic Regression, Random Forest, SVM, ResNet-20 and Resnet-30 result can be shown in Table 4.

Table 4. The technique comparison results in TSL

Alogorithm	Images/Class	Testing Set	Epochs	Maximum Accuracy
Naïve Bayes	7	1	50	41.26
Decision Tree	7	1	50	71.30
KNN	7	1	50	81.72
Logistic Regression	7	1	50	86.76
Random Forest	7	1	50	84.30
SVM	7	1	50	89.42
ResNet-20	7	1	50	92.86
ResNet-30	7	1	50	93.33
Propose	7	1	50	99.14

Table 4 showed the technique comparison result, it has seen that TSR-2DCNN has a great accuracy more than other algorithms, and it is accuracy of 99.14. Next is ResNet-30 accuracy of 93.33, next is ResNet-20 accuracy of 92.86 and SVM accuracy of 89.42, respectively.

7. Conclusion

This research modifies TSR-2DCNN based on Kit-A02. 7 gestures of TSL are dataset and evaluate the accuracy of TSR. The experiment shows accuracy value at 0.9914 and loss value at 0.03537. It implies that modification model can have increased effectiveness in recognition. Total training time is 2hr 4 min 54sec and higher when repetitive training is performed. Moreover, it can diminish the problem of crashing system, overload and Kit-A02 reboot. In future work, the researcher will compare with LSTM and RNN algorithm and enhance feature extraction to investigate all components before applying in real-time.

References

[1]. Alani, A. A., & Cosma, G. (2021). ArSL-CNN: a convolutional neural network for Arabic sign language gesture recognition. *Indonesian Journal of Electrical Engineering and Computer Science, 22*(2), 1096-1107.

[2]. Camgoz, N. C., Koller, O., Hadfield, S., & Bowden, R. (2020). Sign language transformers: Joint end-to-end sign language recognition and translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10023-10033).

[3]. Cui, R., Liu, H., & Zhang, C. (2019). A deep neural framework for continuous sign language recognition by iterative training. *IEEE Transactions on Multimedia*, *21*(7), 1880-1891.

[4]. Adeyanju, I. A., Bello, O. O., & Adegboye, M. A. (2021). Machine learning methods for sign language recognition: A critical review and analysis. *Intelligent Systems with Applications*, *12*, 200056.

[5]. Elpeltagy, M., Abdelwahab, M., Hussein, M. E., Shoukry, A., Shoala, A., & Galal, M. (2018). Multi-modality-based Arabic sign language recognition. *IET Computer Vision*, *12*(7), 1031-1039.

[6]. Kodama, T., Koyama, T., & Saitoh, T. (2017, September). Kinect sensor based sign language word recognition by mutli-stream HMM. In *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)* (pp. 94-99). IEEE.

[7]. Deriche, M., Aliyu, S. O., & Mohandes, M. (2019). An intelligent arabic sign language recognition system using a pair of LMCs with GMM based classification. *IEEE Sensors Journal*, *19*(18), 8067-8078.

[8]. Liu, Q., Qian, G., Meng, W., Ai, Q., Yin, C., & Fang, Z. (2019). A new IMMU-based data glove for hand motion capture with optimized sensor layout. *International Journal of Intelligent Robotics and Applications*, *3*(1), 19-32.

[9]. Bhuyan, M. K., Talukdar, A. K., Gupta, P., & Laskar, R. H. (2020, August). Low Cost Data Glove for Hand Gesture Recognition by Finger Bend Measurement. In *2020 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)* (pp. 25-31). IEEE.

[10]. Qi, J., Jiang, G., Li, G., Sun, Y., & Tao, B. (2020). Surface EMG hand gesture recognition system based on PCA and GRNN. *Neural Computing and Applications*, *32*(10), 6343-6351.

[11]. Wu, J., Sun, L., & Jafari, R. (2016). A wearable system for recognizing American sign language in real-time using IMU and surface EMG sensors. *IEEE journal of biomedical and health informatics*, *20*(5), 1281-1290.

[12]. Yang, X., Chen, X., Cao, X., Wei, S., & Zhang, X. (2017). Chinese Sign Language Recognition Based on an Optimized Tree-Structure Framework. *IEEE journal of biomedical and health informatics*, *21*(4), 994-1004.

[13]. Amrutha, K., & Prabu, P. (2021, February). ML Based Sign Language Recognition System. In *2021 International Conference on Innovative Trends in Information Technology (ICITIIT)* (pp. 1-6). IEEE.

[14]. Fontaine, J., Shahid, A., Elsas, R., Seferagic, A., Moerman, I., & De Poorter, E. (2020, November). Multi-band sub-GHz technology recognition on NVIDIA's Jetson Nano. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)* (pp. 1-7). IEEE.

[15]. Cheok, M. J., Omar, Z., & Jaward, M. H. (2019). A review of hand gesture and sign language recognition techniques. *International Journal of Machine Learning and Cybernetics*, *10*(1), 131-153.

[16]. Zheng, Y., Zhao, C., Lei, Y., & Chen, L. (2020, December). Embedded Radio Frequency Fingerprint Recognition Based on A Lightweight Network. In *2020 IEEE 6th International Conference on Computer and Communications (ICCC)* (pp. 1386-1392). IEEE.

[17]. Jiang, S., Li, L., Xu, H., Xu, J., Gu, G., & Shull, P. B. (2020). Stretchable e-Skin Patch for Gesture Recognition on the Back of the Hand. *IEEE Transactions on Industrial Electronics*, *67*(1), 647.

[18]. Jasm, D. A., Hamad, M. M., & Alrawi, A. T. H. (2020). Deep image mining for convolution neural network. *Indones. J. Electr. Eng. Comput. Sci*, *20*, 347-352.

[19]. Basulto-Lantsova, A., Padilla-Medina, J. A., Perez-Pinal, F. J., & Barranco-Gutierrez, A. I. (2020, January). Performance comparative of OpenCV Template Matching method on Jetson TX2 and Jetson Nano developer kits. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0812-0816). IEEE.

[20]. NVIDIA. (2019). Jetson Nano Developer Kit. Retrieved from : https://th.mouser.com/pdfDocs/Jetson_Nano_Developer_Kit_User_Guide.pdf. [accessed: 21 August 2021].

[21]. NVIDIA.(2021). Nvidia Jetson Nano – Bringing AI to Millions of New Devices at the Edge. Retrieved from: http://static6.arrow.com/aropdfconversion/dcf9bb2703550b10cdaca1481ba59766c69be7a0/final-product-brief-for-jetson-nano-bringing-original.pdf. [accessed: 25 August 2021].

[22]. Dhall, I., Vashisth, S., & Aggarwal, G. (2020, January). Automated hand gesture recognition using a deep convolutional neural network model. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 811-816). IEEE.

[23]. Chen, H. K., & Chuang, C. C. (2020, November). A Gesture Controlled Music Playback System Using Convolutional Neural Network. In *2020 International Symposium on Computer, Consumer and Control (IS3C)* (pp. 13-16). IEEE.

[24]. Bird, J. J., Ekárt, A., & Faria, D. R. (2020). British sign language recognition via late fusion of computer vision and leap motion with transfer learning to american sign language. *Sensors*, *20*(18), 5151.

[25]. Shahriar, S., Siddiquee, A., Islam, T., Ghosh, A., Chakraborty, R., Khan, A. I., ... & Fattah, S. A. (2018, October). Real-time american sign language recognition using skin segmentation and image category classification with convolutional neural network and deep learning. In *TENCON 2018-2018 IEEE Region 10 Conference* (pp. 1168-1171). IEEE.

[26]. Rahmat, R. F., Chairunnisa, T., Gunawan, D., & Sitompul, O. S. (2016, August). Skin color segmentation using multi-color space threshold. In *2016 3rd international conference on computer and information sciences (ICCOINS)* (pp. 391-396). IEEE.

[27]. Hamed, A., Belal, N. A., & Mahar, K. M. (2016, February). Arabic sign language alphabet recognition based on HOG-PCA using microsoft kinect in complex backgrounds. In *2016 IEEE 6th international conference on advanced computing (IACC)* (pp. 451-458). IEEE.

[28]. Dhall, I., Vashisth, S., & Aggarwal, G. (2020, January). Automated hand gesture recognition using a deep convolutional neural network model. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 811-816). IEEE.

[29]. Basulto-Lantsova, A., Padilla-Medina, J. A., Perez-Pinal, F. J., & Barranco-Gutierrez, A. I. (2020, January). Performance comparative of OpenCV Template Matching method on Jetson TX2 and Jetson Nano developer kits. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0812-0816). IEEE.