symmetry

MDPI

Article

Multi-Stroke Thai Finger-Spelling Sign Language Recognition System with Deep Learning

Thongpan Pariwat and Pusadee Seresangtakul *

Natural Language and Speech Processing Laboratory, Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen 40002, Thailand; thongpan.par@kkumail.com
* Correspondence: pusadee@kku.ac.th; Tel.: +66-80-414-0401

Abstract: Sign language is a type of language for the hearing impaired that people in the general public commonly do not understand. A sign language recognition system, therefore, represents an intermediary between the two sides. As a communication tool, a multi-stroke Thai finger-spelling sign language (TFSL) recognition system featuring deep learning was developed in this study. This research uses a vision-based technique on a complex background with semantic segmentation performed with dilated convolution for hand segmentation, hand strokes separated using optical flow, and learning feature and classification done with convolution neural network (CNN). We then compared the five CNN structures that define the formats. The first format was used to set the number of filters to 64 and the size of the filter to 3×3 with 7 layers; the second format used 128 filters, each filter 3×3 in size with 7 layers; the third format used the number of filters in ascending order with 7 layers, all of which had an equal 3×3 filter size; the fourth format determined the number of filters in ascending order and the size of the filter based on a small size with 7 layers; the final format was a structure based on AlexNet. As a result, the average accuracy was 88.83%, 87.97%, 89.91%, 90.43%, and 92.03%, respectively. We implemented the CNN structure based on AlexNet to create models for multi-stroke TFSL recognition systems. The experiment was performed using an isolated video of 42 Thai alphabets, which are divided into three categories consisting of one stroke, two strokes, and three strokes. The results presented an 88.00% average accuracy for one stroke, 85.42% for two strokes, and 75.00% for three strokes.

Keywords: TFSL recognition system; deep learning; semantic segmentation; optical flow; complex background

Citation: Pariwat, T.; Seresangtakul, P. Multi-Stroke Thai Finger-Spelling Sign Language Recognition System with Deep Learning. *Symmetry* **2021**, *13*, 262. https://doi.org/10.3390/sym13020262

Academic Editor: Peng-Yeng Yin
Received: 11 January 2021
Accepted: 31 January 2021
Published: 4 February 2021

1. Introduction

Hearing-impaired people around the world use sign language as their medium for communication. However, sign language is not universal, with one hundred varieties used around the world [1]. One of the most widely used types of sign language is American Sign Language (ASL), which is used in the United States, Canada, West Africa, and Southeast Asia and influences Thai Sign Language (TSL). Typically, global sign language is divided into two forms: gesture language and finger-spelling. Gesture language or sign language involves the use of hand gestures, facial expressions, and the use of mouths and noses to convey meanings and sentences. This type is used for communication between deaf people in everyday life, focusing on terms such as eating, ok, sleep, etc. Finger-spelling sign language is used for spelling letters related to the written language with one's fingers and is used to spell people' names, places, animals, and objects.

ASL is the foundation of Thai finger-spelling sign language (TFSL).The TFSL was invented in 1953 by Khunying Kamala Krairuek using American finger-spelling as a prototype to represent the 42 Thai consonants, 32 vowels, and 6 intonation marks [2]. All forty-two Thai letters can be presented with a combination of twenty-five hand gestures. For this purpose, the number signs are combined with alphabet signs to create additional

meanings. For example, the K sign combined with the 1 sign (K+1) meaning 'ข' (/k^h/). Consequently, TFSL strokes can be organized into three groups (one-stroke, two-stroke, and three-stroke groups) with 15 letters, 24 letters, and 3 letters, respectively. However, 'ฃ'(/k^h/) and 'ฅ'(/k^h/) have not been used since Thailand's Royal Institute Dictionary announced their discontinuation [3], and are no longer used in Thai finger-spelling sign language.

This study is part of a research series on TFSL recognition that focuses on one stroke performed by a signer standing in front of a blue background. Global and local features using support vector machine (SVM) on the radial basis function (RBF) kernel were applied and were able to achieve 91.20% accuracy [4]; however, there were still problems with highly similar letters. Therefore, a new TFSL recognition system was developed by applying pyramid histogram of oriented gradient (PHOG) and local features with k-nearest neighbors (KNN), providing an accuracy up to 97.6% [3]. However, past research has been conducted only using one-stroke, 15-character TFSL under a simple background, which does not cover all types of TFSL. There is also research on TFSL recognition systems that uses a variety of techniques, as shown in Table 1.

Table 1. Research on Thai finger-spelling sign language (TFSL) recognition systems.

Researchers	No. of Sign	Method	Dataset	Accuracy (%)
Adhan and Pintavirooj [5]	42	Black glove with 6 sphere marker, geometric invariant and ANN	1050	96.19
Saengsri et al. [6]	16	Glove with motion tracking	(N/A)	94.44
Nakjai and Kantanyakul [2]	25	CNN	1375	91.26
Chansri and Srinonchat [7]	16	HOG and ANN	320	83.33
Silanon [8]	16	HOG and ANN	2100	78.00

The study of TFSL recognition systems for practical applications requires using a variety of techniques, since TFSL has multi-stroke gestures and combines hand gestures to convey letters as well as complex background handling and a wide range of light volumes.

Deep learning is a tool that is increasingly being used in sign language recognition [2,9,10], face recognition [11], object recognition [12], and others. This technology is used for solving complex problems such as object detection [13], image segmentation [14], and image recognition [12]. With this technique, the feature learning method is implemented instead of feature extraction. Convolution neural network (CNN) is a learning feature process that can be applied to recognition and can provide high performance. However, deep learning requires much data for training, and such data could use simple background or complex background images. In the case of a picture with a simple background, the object and background color are clearly different. Such an image can be used as training data without cutting out the background. On the other hand, a complex background image features objects and backgrounds that have similar colors. Cutting out the background image will obtain input images for training with deep learning, featuring only objects of interest without distraction. This can be done using the semantic segmentation method, which uses deep learning to apply image segmentation. This method requires labeling objects of interest to separate them from the background. Autonomous driving is one application of semantic segmentation used to identify objects on the road, and can also be used for a wide range of other applications.

This study is based on the challenges of the TFSL recognition system. There are up to 42 Thai letters that involve spelling with one's fingers—i.e., spelling letters with a combination of multi-stroke sign language gestures. The gestures for spelling several letters, however, are similar, so a sign language recognition system under a complex background should instead be used. There are many studies about the TFSL recognition system. However, none of them study the TFSL multi-stroke recognition system using the vision-based technique with complex background. The main contributions of this research are the application of a new framework for a multi-stroke Thai finger-spelling sign language recognition system to videos under a complex background and a variety of light intensities by

separating people's hands from the complex background via semantic segmentation methods, detecting changes in the strokes of hand gestures with optical flow, and learning features with CNN under the structure of AlexNet. This system supports recognition that covers all 42 characters in TFSL.

The proposed method focuses on developing a multi-stroke TFSL recognition system with deep learning that can act as a communication medium between the hearing impaired and the general public. Semantic segmentation is then applied to hand segmentation for complex background images, and optical flow is used to separate the strokes of sign language. The processes of feature learning and classification use CNN.

2. Review of Related Literature

Research on sign language recognition systems has been developed in sequence with a variety of techniques to obtain a system that can be used in the real world. Deep learning is one of the most popular methods effectively applied to sign language recognition systems. Nakjai and Katanyukul [2] used CNN to develop a recognition system for TFSL with a black background image and compared 3-layer CNN, 6-layer CNN, and HOG, finding that the 3-layer CNN with 128 filters on every layer had an average precision (mAP) of 91.26%. Another study that applied deep learning to a sign language recognition system was published by Lean Karlo S. et al. [9], who developed an American sign language recognition system with CNN under a simple background. The dataset was divided into 3 groups: alphabet, number, and static word. The results show that alphabet recognition had an average accuracy of 90.04%, number recognition had an accuracy average of 93.44%, and static word recognition had an average accuracy of 97.52%. The total average of the system was 93.67%. Rahim et al. [15] applied deep learning to a non-touch sign word recognition system using hybrid segmentation and CNN feature fusion. This system used SVM to recognize sign language. The research results under a real-time environment provided an average accuracy of 97.28%.

Various sign language studies aimed to develop a vision base by using, e.g., image and video processing, object detection, image segmentation, and recognition systems. Sign language recognition systems using vision can also be divided into two types of backgrounds: simple backgrounds [3,4,9,16] and complex backgrounds [17–19]. A simple background entails the use of a single color such as green, blue, or white. This can help hand segmentation work more easily, and the system can recognize accuracy at a high level. Pariwat et al. [4] used sign language images on a blue background while the signer wore a black jacket in a TFSL recognition system with SVM on RBF, including global and local features. The average accuracy was 91.20%. Pariwat et al. [3] also used a simple background in a system that was developed by combining PHOG and local features with KNN. This combination enhanced the average accuracy to levels as high as 97.80%. Anh et al. [10] presented a Vietnamese language recognition system using deep learning in a video sequence format. A simple background, like a white background, was used in training and testing, providing an accuracy of 95.83%. Using a simple background makes it easier to develop a sign language recognition system and provides high-accuracy results, but this method is not practical due to the complexity of the backgrounds in real-life situations.

Studying sign language recognition systems with complex backgrounds is challenging. Complex backgrounds consist of a variety of elements that are difficult to distinguish, which requires more complex methods. Chansri et al. [7] developed a Thai sign language recognition system using Microsoft Kinect to help locate a person's hand in a complex background. This study used the fusion of depth and color video, HOG features, and a neural network, resulting in accuracy of 84.05%. Ayman et al. [17] proposed the use of Microsoft Kinect for Arabic sign language recognition systems with complex backgrounds with histogram of oriented gradients—principal component analysis (HOG-PCA) and SVM. The resulting accuracy value was 99.2%. In summary, research on sign language recognition systems using vision-based techniques remains a challenge for real-world applications.

3. The Proposed Method

This proposed method is a development of the multi-stroke TFSL recognition system covering 42 letters. The system consists of four parts: (1) creating a SegNet model via semantic segmentation using dilated convolutions; (2) creating a hand-segmented library with a SegNet model; (3) TFSL model creation with CNN for the TFSL model; and (4) classification processes using optical flow for hand stroke detection and using the CNN model for classification. Figure 1 shows an overview of the system.

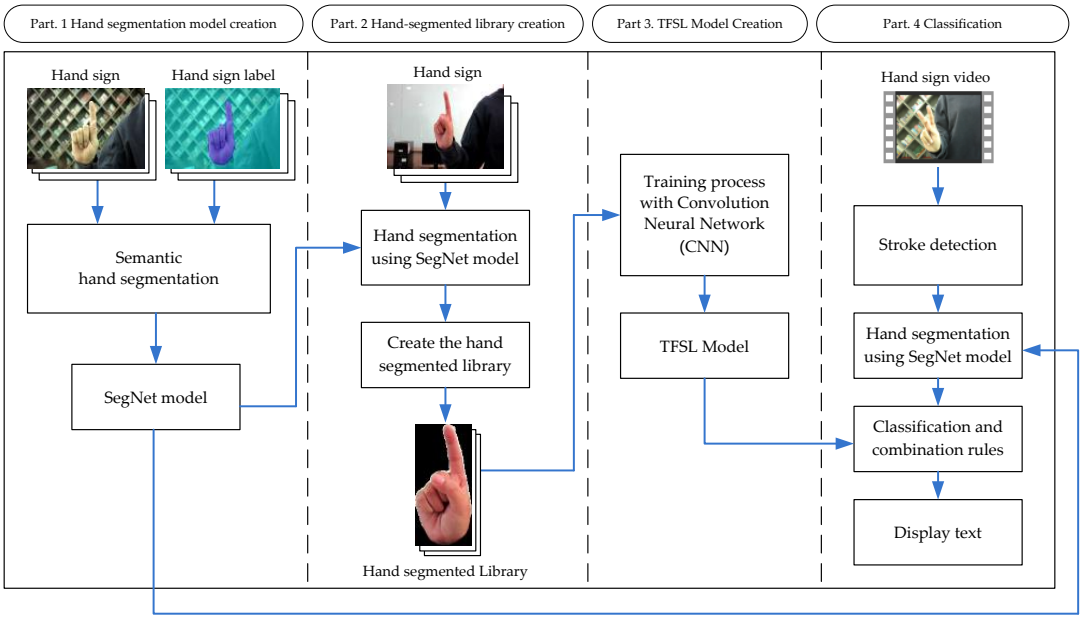


Figure 1. Multi-stroke Thai finger-spelling sign language recognition framework.

3.1. Hand Segmentation Model Creation

Segmentation is one of the most important processes for recognition systems. In particular, sign language recognition systems require the separation of the hands from the background image. Complex backgrounds are a challenging task for hand segmentation because they feature a wide range of colors and lighting. Semantic segmentation has been effectively applied to complex background images and it is used to describe images at a pixel level with a class label [20]. For example, in an image containing people, trees, cars, and signs, semantic segmentation can find separate labels for the objects depicted in the image, and it is applied to autonomous vehicles, robotics, human–computer interactions, etc.

Dilated convolution is used in segmentation tasks and has consistently improved accuracy performance [21]. Dilated convolution provides a way to exponentially increase the receptive view (global view) of the network and provide linear parameter accretion. In general, dilated convolution includes the application of an input with spacing defined by the dilation rate. For example, in Figure 2, a 1-dilated convolution (a) refers to the normal convolution with a 3×3 receptive field, a 2-dilated (b) refers to one-pixel spacing with a size of 7×7, and a 4-dilated (c) refers to a 3-pixel spacing receptive field with a size of 15×15. The red dot represents the 3×3 input filters, and the green area is the receptive area of these inputs.

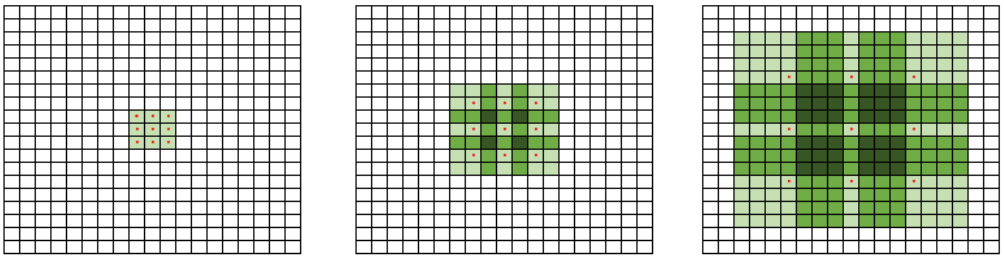


Figure 2. Dilated convolution on a 2D image: (**a**) The normal convolution uses a 3 × 3 receptive field; (**b**) one-pixel spacing has 7 × 7; (**c**) and three-pixel spacing uses a receptive field size of 15 × 15.

This study employs semantic segmentation using dilated convolutions to create a hand-segmented library. The process for creating a segmented library is as shown in Figure 3. First, create image files and image labels for hand signs as training data. Second, create a semantic segmentation network model (SegNet model). The three layers of convolution are as follows: dilated convolution, batch normalization, and rectified linear unit (ReLU), with the final layer used for softmax and pixel classification. Next, once data training is completed, the results of the SegNet model are ready to be used in the next step. The SegNet model that was created is applied in the process of creating a hand-segmented library.

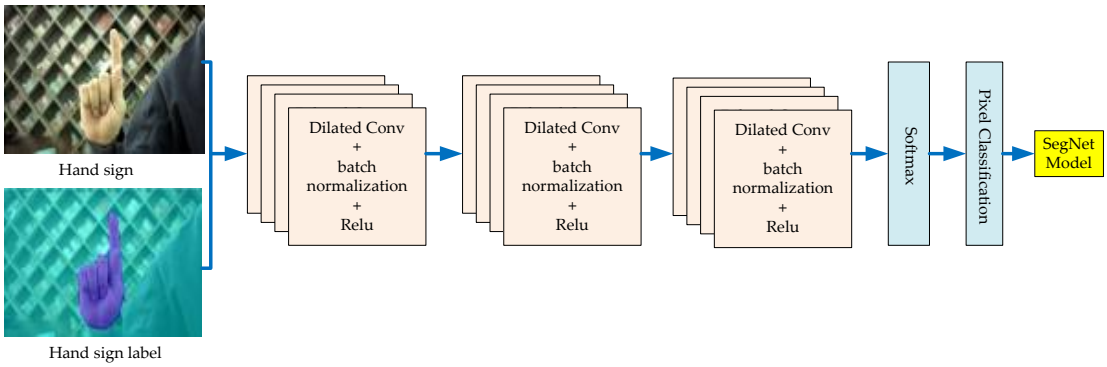


Figure 3. Hand segmentation model creation via semantic segmentation.

3.2. Hand-Segmented Library Creation

Creating a hand-segmented library is done as preparation for training, with the input hand sign image used to create a library consisting of 25 gestures of TFSL. Each gesture has 5000 images for a total of 125,000 images. This process includes the following processes, as shown in Figure 4. The first process is to engage in semantic segmentation by applying the SegNet model from the previous process to segment the hands and background image. The second process is to label the position of the hand as a binary image. The third process is to blur the Gaussian filter method to reduce the noise of the image and remove small space object from binary image, leaving only areas of wide space. The fourth process sorts the areas in ascending order, selects the largest space (area of the hand), and creates a bounding box at the position of the hand to frame only the features of interest. The last process is to crop the image by hand by applying a cropped binary image to the input image to remove only the interesting part and then resize the image to 150 × 250 pixels. Examples of images from the hand-segmented library are shown in Figure 5.

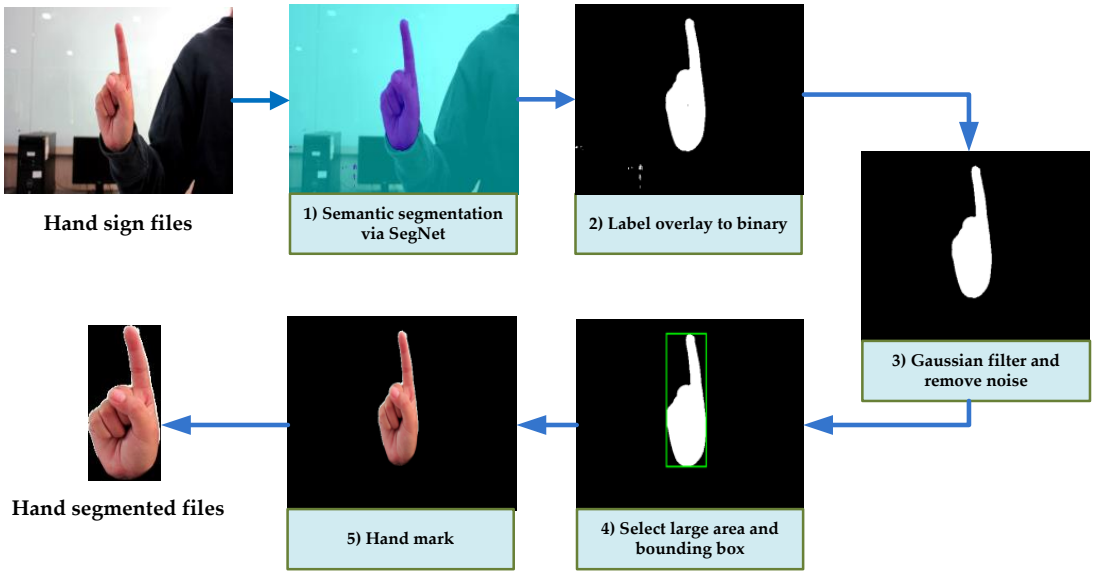


Figure 4. The process of creating a hand-segmented library.

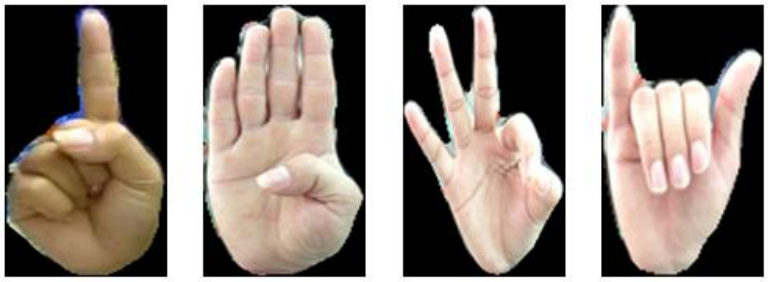


Figure 5. Examples of the hand-segmented library files.

3.3. TFSL Model Creation

This process uses data training methods to learn features and create a CNN classification model, as shown in Figure 6. The training starts with introducing data from a hand-segmented library into deep learning via CNN, which consists of multiple layers in the feature detection, processes, each with convolution (Conv), ReLU, and pooling [22].

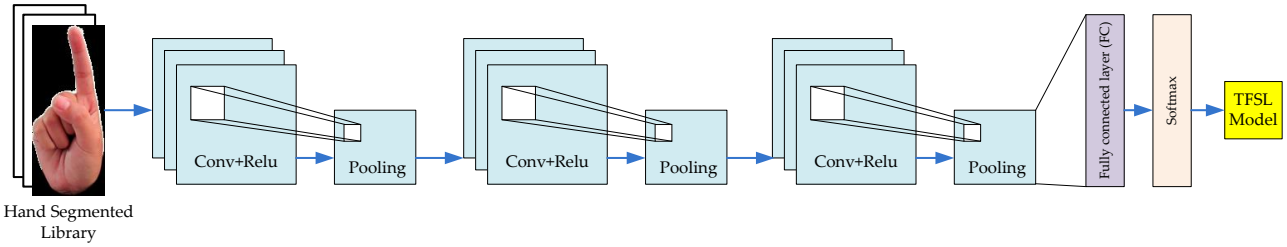


Figure 6. Training process with convolution neural network (CNN).

Three processes are repeated for each cycle of layers, with each layer learning to detect different features. Convolution is a layer that learns features from the input image. For convolution, small squares of input data are used to learn the image features by preserving the relationship between pixels. The image matrix and filter or kernel are mathematically produced by the dot product, as shown in Figure 7.

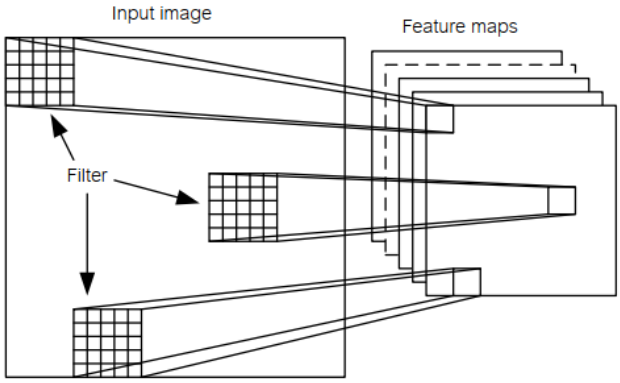


Figure 7. Learning features with a convolution filter [23].

ReLU is an activation function that allows an algorithm to work faster and more effectively. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value, as presented in Figures 8 and 9. Thus, ReLU can be written as Equation (1):

$$f(x) = \max(0, x) \quad (1)$$

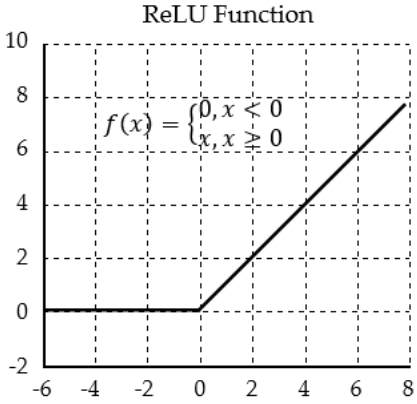


Figure 8. The ReLU function.

19	84	-125	9	52
56	-78	12	52	-206
7	22	-45	12	96
75	-8	10	63	126
32	-3	85	95	44
6	80	178	95	-78

19	84	0	9	52
56	0	12	52	0
7	22	0	12	96
75	0	10	63	126
32	0	85	95	44
6	80	178	95	0

Figure 9. An example of the values after using the ReLU function [20].

Pooling resizes the data to a smaller size, with the details of the input remaining intact. It also has the advantage of increasing the sensitivity to calculations and solving the problems of overfitting. There are two types of pooling layers, max and average pooling, as illustrated in Figure 10.

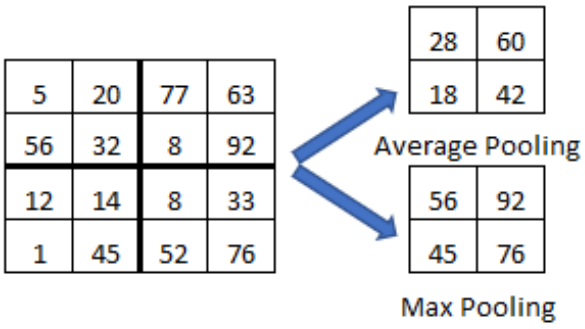


Figure 10. An example of average and max pooling [23].

The final layer of the CNN architecture is the classification layer, consisting of 2 sub-layers: the fully connected (FC) and softmax layers. The fully connected layer will use the feature map matrix in the form of a vector derived from the feature detection process to create predictive models. The softmax function layer provides the classification output.

3.4. Classification

In the classification process, video files showing isolated TFSL were used as the input. The video input files consisted of one stroke, two strokes, and three strokes. These files were taken with a digital camera and required the signer to act with the right hand in a black jacket standing in front of a complex background. The whole process of testing involved the following steps: motion segmentation by optical flow, splitting the strokes of the hand gestures, hand segmentation via the SegNet model, classification, combining signs, and displaying the text, as shown in Figure 11.

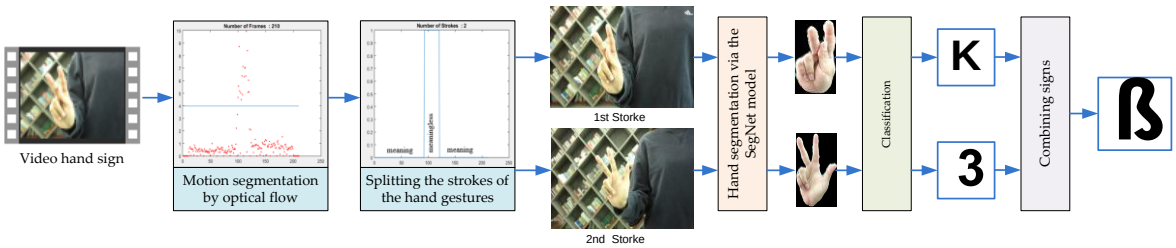


Figure 11. Classification and display text process.

Motion segmentation used Lucas and Kanade's optical flow calculation method, which is a method of tracking the whole image using the pyramid algorithm. The track starts with the top layer of the pyramid and runs down to the bottom. Motion segmentation was performed by optical flow using the orientation and magnitude of the vector. Calculating the orientation and magnitude of the optical flow was accomplished using frame-by-frame calculations considering 2D motion. In each frame, the angle of the motion at each point varied, depending on the magnitude and direction of the motion observed from the vector relative to the axis (x, y). The direction of motion can be obtained from Equation (2) [24]:

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) \quad (2)$$

where θ is the direction of vector $v = [x,y]^T$, x is coordinate x, and y is coordinate y in the range of $-\frac{\pi}{2} + \pi\frac{b-1}{B} \leq \theta < -\frac{\pi}{2} + \pi\frac{b}{B}$ where b is the number of block and B is the amount of bins.

The magnitude of optical flow calculates the vector length by using the linear equation to find the length of each vector between the previous frame and the current frame. The magnitude can be calculated from the equation below [24]:

$$m = \sqrt{x^2 + y^2} \tag{3}$$

where m is the magnitude, x is coordinate x, and y is coordinate y.

In this research, motion was distinguished by finding the mean of all magnitudes within each frame of the split hand sign. The mean of the magnitude was high when the hand signals were changed, as shown in Figure 12.

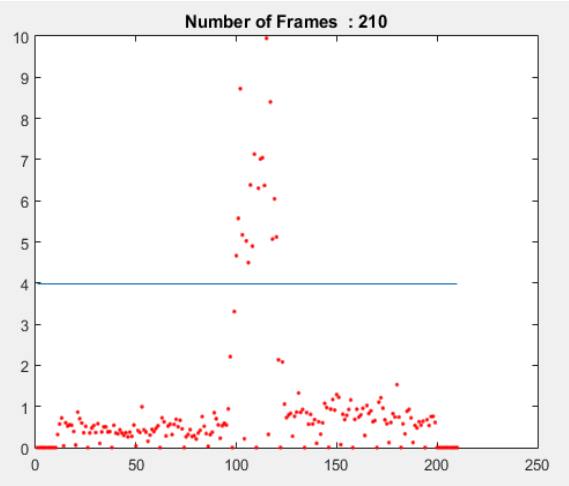


Figure 12. The mean value in each frame of magnitude of the video files.

Splitting the stroke of the hand was achieved by comparing each stroke with the threshold value. Testing the various threshold values, it was found that a magnitude value of 1–3 means slight movement, and a value equal to 4 or more indicates a change in the stroke of the hand signal. The threshold value in this experiment was considered as 4. If the mean value of magnitude in each frame was lower than the threshold value, then s = 0 (meaning), but if the value was more than the threshold value, then s = 1 (meaningless), as shown in Figure 13. After that, 10 frames between the start and the end of s = 0 (meaning) represent the hand sign.

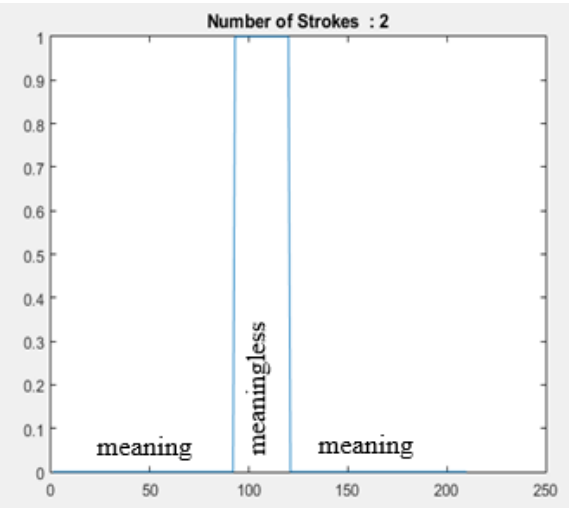


Figure 13. The strokes of the hand sign.

After extracting the image from the hand stroke separation, the next step was the hand segmentation process. This process involves the separation of hands from the background using the SegNet model, as illustrated in Figure 14.

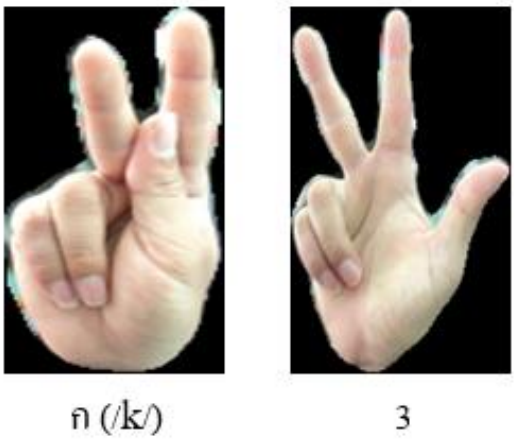


Figure 14. Image of hand segmentation after splitting the strokes of the hand.

The next step was to classify the hand segmentation images with the CNN model to predict 10 frames of each hand sign. The system votes on the results of the prediction in descending order and selects the most predictable hand sign. As shown in Figure 11, when passing the classification process, the results can be predicted as the K sign and the 3 sign. After that, the signs are combined and then translated into Thai letters. The rules for the combination of signs are shown in Table 2.

Table 2. Combination rules for TFSL.

No.	Rule	Alphabet	No.	Rule	Alphabet
1	K	ก (/k/)	22	R	ร (/r/)
2	K + 1	ข (/k^h/)	23	W	ว (/w/)
3	K + 2	ค (/k^h/)	24	D	ด (/d/)
4	K + 3	ฆ (/k^h/)	25	D + 1	ฎ (/d/)
5	T	ต (/t/)	26	F	ฟ (/f/)
6	T + 1	ถ (/t^h/)	27	F + 1	ฝ (/f/)
7	T + 2	ฐ (/t^h/)	28	L	ล (/l/)
8	T + 3	ฒ (/t^h/)	29	L + 1	ฬ (/l/)
9	T + 4	ท (/t^h/)	30	J1 + J2	จ (/t͡ɕ/)
10	T + 5	ฏ (/ t̚ /)	31	Y	ย (/j/)
11	S	ส (/s/)	32	Y + 1	ญ (/j/)
12	S + 1	ศ (/s/)	33	M	ม (/m/)
equ	S + 2	ษ (/s/)	34	N	น (/n/)
14	S + Q	ซ (/s/)	35	N + 1	ณ (/n/)
15	P	พ (/p^h/)	36	N + G	ง (/ŋ/)
16	P + 1	ป (/p/)	37	T + H	ท (/t^h/)
17	P + 2	ผ (/p^h/)	38	T + H + 1	ธ (/t^h/)
18	P + 3	ภ (/p^h/)	39	C + H	ฉ (/t͡ɕ^h/)
19	H	ห (/h/)	40	C + H + 1	ช (/t͡ɕ^h/)
20	H + 1	ฮ (/h/)	41	C + H + 2	ฌ (/t͡ɕ^h/)
21	B	บ (/b/)	42	A	อ (/ʔ/)

4. Experiments and Results

4.1. Data Collection

Data collection for this research is divided into two types: images from 6 locations for training, and videos for testing at 3 locations (Library, Computer Lab2, and Office1), which take a slightly different view of the camera. They were taken with a digital camera

and required the signer to use his or her right hand to make a hand signal while standing in front of a complex background at the 6 locations presented in Figure 15.

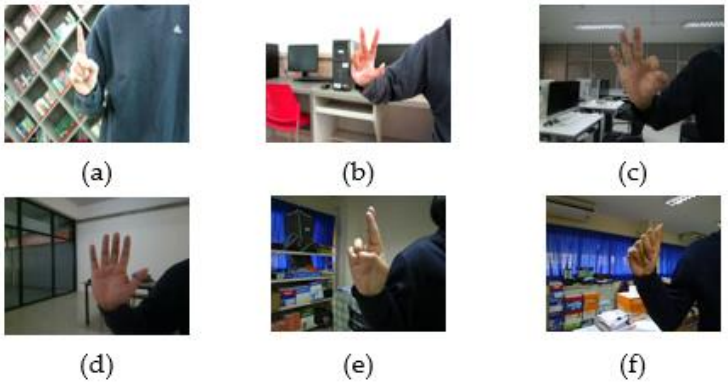


Figure 15. The six locations of data collection (**a**) Library; (**b**) Computer Lab1; (**c**) Computer Lab2; (**d**) Canteen; (**e**) Office1; and (**f**) Office2.

The hand sign images used for the training were divided into 2 groups: the training data for the SegNet model and the training data for the CNN model, with 25 signs in total. The training data for the SegNet model used original images and image labels, 320 × 240 pixels in size, totaling 10,000 images. The second group contained images that were hand-segmented via semantic segmentation and were contained in the hand-segmented library. These images were 150 × 250 pixels in size. The training dataset contained 25 signs, as presented in Figure 16, each with 5000 images for a total of 125,000 images.

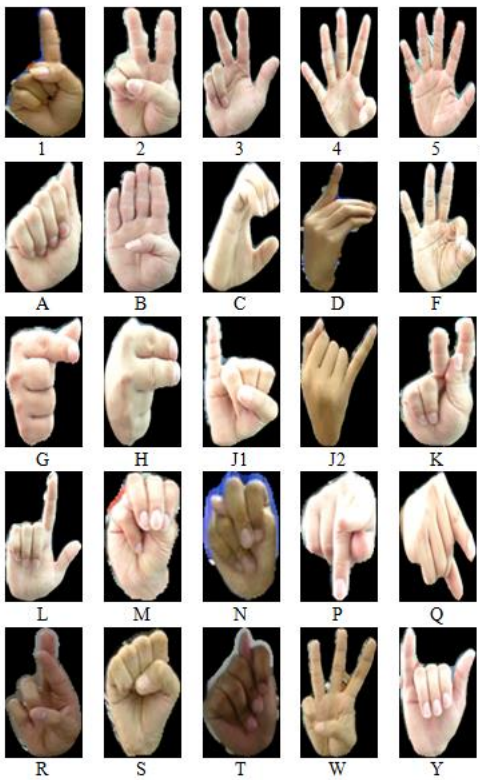


Figure 16. Twenty-five hand signs for 42 Thai letters.

The video used in the testing process was isolated TFSL featuring 42 Thai letters. These videos were taken from a hand signer group consisting of 4 people, and each letter was recorded 5 times, totaling (42 × 4 × 5) 840 files.

4.2. Experimental Results

The experiments in this study were divided into three parts. The first part involved the evaluation to measure performance using an intersection over union areas (IoU); the second part was the experiment used to determine the efficiency of the CNN model; and the last part was the testing of the multi-stroke TFSL recognition system.

4.2.1. Intersection Over Union Areas (IoU)

IoU is a statistical method that uses two data consistency measurements between a predicted bounding box and ground truth by dividing the overlapping areas between the prediction and ground truth by a union area between the prediction and ground truth, as Equation (4) and shown in Figure 17 [18]. A high-value IoU (value closer to 1) refers to a bounding box with high accuracy. The 0.5 threshold is the accuracy that determines whether the predicted bounding box IoU is accurate, as shown in Figure 18. According to the Standard Pascal Visual Object Classes Challenge 2007, acceptable IoU values must be greater than 0.5 [13].

$$\text{IoU} = \frac{\text{Area of overlap between bounding boxes}}{\text{Area of union between bounding boxes}}. \quad (4)$$

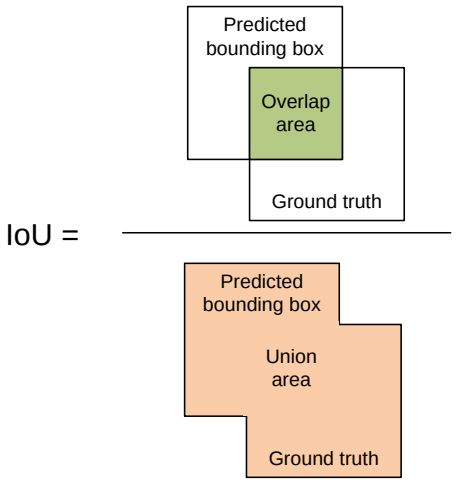


Figure 17. The intersection over union areas (IoU) [25].

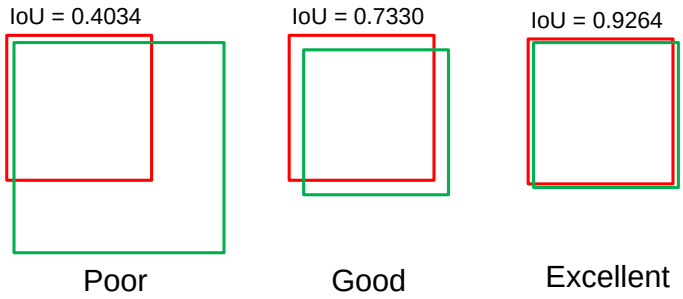


Figure 18. IoU is evaluated as follows: the left IoU is 0.4034, which is poor; the middle IoU = 0.7330, which is good; and the high IoU = 0.9264, which is excellent [13].

The research used sign language gestures on a complex background along with label images, all of which were 320 × 240 pixels in size. A total of 10,000 images passed

through semantic segmentation training using dilated convolutions. The results of the dilation rate configuration test and a variety of convolutions can be structured according to the following details. The structure of the training process consists of five blocks of convolution. Each block includes 32 convolution filters (3 × 3 in size), with different dilation factors, batch normalization, and ReLU. The dilations are 1, 2, 4, 8, 16, and 1, respectively. The training option uses MaxEpochs = 500 and MiniBatchSize = 64. According to the IoU performance assessment, the average IoU was 0.8972, which is excellent.

4.2.2. Experiments on the CNN Models

Experiments to determine the effectiveness of the CNN models in this research used the 5-fold cross validation method. All input hand sign images used in this experiment contained 125,000 images from the hand-segmented library, which consisted of 25 finger-spelling images, each with 5000 gestures. This dataset was divided into 5 groups, each with 25,000 images. Each round of 5-fold cross validation used 100,000 images for training and 25,000 images for testing. This experiment compared the effectiveness of the configuration, the number of filters, and the sizes of five different CNN filters.

The first format used seven layers of 64 filters and a 3 × 3 filter size. The second format set the filter size to 128 and the size of the filter to 3 × 3 with 7 layers. The third format sorted the number of filters in ascending order up to 7 layers, starting from 2, 5, 10, 20, 40, 80, and 160, respectively, and all filter sizes were 3 × 3. The fourth format determined the number of filters in ascending order and set the sizes of the filter from large to small with 7 layers (the number of filters/filter size; 4/11 × 11, 8/5 × 5, 16/5 × 5, 32/3 × 3, 64/3 × 3, 128/3 × 3, 256/3 × 3). The final format was a structure based on AlexNet, which defined the numbers and sizes of the filters as follows (number of filters/filter size): 96/11 × 11, 256/5 × 5, 384/3 × 3, 384/3 × 3, 256/3 × 3, 256/3 × 3 [26]. The structure details are shown in Figure 19.

Format 1

Convolution layer 1	Convolution layer 2	Convolution layer 3	Convolution layer 4	Convolution layer 5	Convolution layer 6	Convolution layer 7	Fully-connected	Softmax
Number of filters 64, filter size 3x3, stride 1x1	Number of filters 64, filter size 3x3, stride 1x1	Number of filters 64, filter size 3x3, stride 1x1	Number of filters 64, filter size 3x3, stride 1x1	Number of filters 64, filter size 3x3, stride 1x1	Number of filters 64, filter size 3x3, stride 1x1	Number of filters 64, filter size 3x3, stride 1x1		
batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization		
ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU		
Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1		

Format 2

Convolution layer 1	Convolution layer 2	Convolution layer 3	Convolution layer 4	Convolution layer 5	Convolution layer 6	Convolution layer 7	Fully-connected	Softmax
Number of filters 128, filter size 3x3, stride 1x1	Number of filters 128, filter size 3x3, stride 1x1	Number of filters 128, filter size 3x3, stride 1x1	Number of filters 128, filter size 3x3, stride 1x1	Number of filters 128, filter size 3x3, stride 1x1	Number of filters 128, filter size 3x3, stride 1x1	Number of filters 128, filter size 3x3, stride 1x1		
batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization		
ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU		
Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1		

Format 3

Convolution layer 1	Convolution layer 2	Convolution layer 3	Convolution layer 4	Convolution layer 5	Convolution layer 6	Convolution layer 7	Fully-connected	Softmax
Number of filters 2, filter size 3x3, stride 1x1	Number of filters 5, filter size 3x3, stride 1x1	Number of filters 10, filter size 3x3, stride 1x1	Number of filters 20, filter size 3x3, stride 1x1	Number of filters 40, filter size 3x3, stride 1x1	Number of filters 80, filter size 3x3, stride 1x1	Number of filters 160, filter size 3x3, stride 1x1		
batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization		
ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU		
Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1		

Format 4

Convolution layer 1	Convolution layer 2	Convolution layer 3	Convolution layer 4	Convolution layer 5	Convolution layer 6	Convolution layer 7	Fully-connected	Softmax
Number of filters 4, filter size 11x11, stride 1x1	Number of filters 8, filter size 5x5, stride 1x1	Number of filters 16, filter size 5x5, stride 1x1	Number of filters 32, filter size 3x3, stride 1x1	Number of filters 64, filter size 3x3, stride 1x1	Number of filters 128, filter size 3x3, stride 1x1	Number of filters 256, filter size 3x3, stride 1x1		
batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization	batchNormalization		
ReLU	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU		
Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1	Max-pooling size 2x2, stride 1x1		

Format 5

Convolution layer 1	Convolution layer 2	Convolution layer 3	Convolution layer 4	Convolution layer 5	Convolution layer 6	Fully-connected+ReLU	Fully-connected+ReLU	Fully-connected	Softmax
Number of filters 96, filter size 11x11, stride 4x4	Number of filters 256, filter size 5x5, stride 1x1	Number of filters 384, filter size 3x3, stride 1x1	Number of filters 384, filter size 3x3, stride 1x1	Number of filters 256, filter size 3x3, stride 1x1	Number of filters 256, filter size 3x3, stride 1x1				
ReLU	ReLU			ReLU	ReLU				
cross-channel normalization	cross-channel normalization	ReLU	ReLU	Max-pooling size 3x3, stride 2x2	Max-pooling size 3x3, stride 2x2				
Max-pooling size 3x3, stride 2x2	Max-pooling size 3x3, stride 2x2			cross-channel normalization	cross-channel normalization				

Figure 19. The five CNN formats for training the model.

The CNN model experiment used the 5-fold cross validation method, and the results are shown in Table 3. This experiment found that the accuracy results of the experiment using the structure based on the AlexNet format were 92.03%. The fourth model of 7 layers using the ascending sorting of the filter number and filter size from a larger size to a smaller size offered an average accuracy of up to 90.04%. Format 3, which uses seven descending filters, has a secondary accuracy average of 89.91%. While the first and second formats use seven static filters, they have 64 and 128 filters, respectively, with an average accuracy of 88.23 and 87.97.

Table 3. Five-fold cross validation for CNN model testing.

Five-Fold Cross Validation	Format 1 Accuracy (%)	Format 2 Accuracy (%)	Format 3 Accuracy (%)	Format 4 Accuracy (%)	Format 5 Accuracy (%)
1st	80.12	80.60	83.43	84.17	87.72
2nd	87.12	85.90	86.28	86.15	88.96
3rd	90.06	90.34	92.44	92.53	93.98
4th	93.84	92.82	95.52	94.87	97.00
5th	90.00	90.20	91.86	92.47	92.50
Average accuracy	88.23	87.97	89.91	90.04	92.03

Comparing the five CNN accuracy results, it was found that the fifth format provided the highest accuracy. Therefore, we used this format for training to create models for the multi-stroke TFSL recognition system, divided into 112,500 training images representing 90% and 12,500 for testing data, representing 10%. The accuracy of the model in the experiment was 99.98%.

4.2.3. The Experiment of the Multi-Stroke TFSL Recognition System

This recognition system focuses on user-independent testing via the isolated TFSL video format. The accuracy of each alphabet was tested 20 times. The experiment was divided into three groups: one stroke, as shown in Table 4; the two-stroke results, which are presented in Table 5; and the three-stroke results, which are shown in Table 6.

Table 4. The results of the one-stroke TFSL recognition system.

No.	Thai Letters	Rule	Correct	Incorrect	Accuracy (%)
1	ก (/k/)	K	13	7	65.00
2	ต (/t/)	T	18	2	90.00
3	ส (/s/)	S	18	2	90.00
4	พ (/p^h/)	P	20	0	100.00
5	ห (/h/)	H	20	0	100.00
6	บ (/b/)	B	17	3	85.00
7	ร (/r/)	R	15	5	75.00
8	ว (/w/)	W	17	3	85.00
9	ด (/d/)	D	20	0	100.00
10	ฟ (/f/)	F	20	0	100.00
11	ล (/l/)	L	20	0	100.00
12	ย (/j/)	Y	20	0	100.00
13	ม (/m/)	M	18	2	90.00
14	น (/n/)	N	11	9	55.00
15	อ (/ʔ/)	A	17	3	85.00
		Average accuracy			88.00

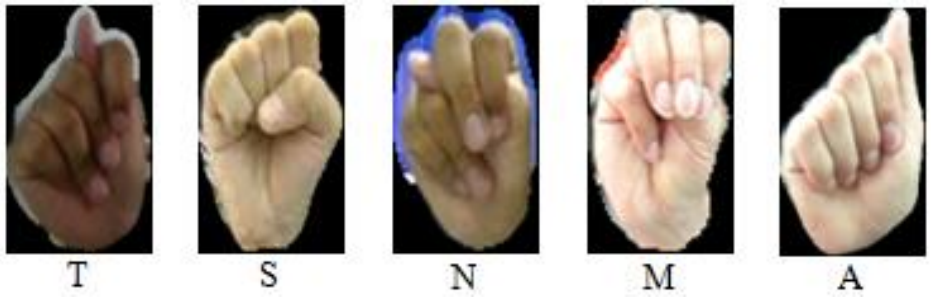
Table 5. The results of the two-stroke TFSL recognition system.

No.	Thai Letters	Rule	Classification			Strokes Detection		
			Correct	Incorrect	Accuracy (%)	Correct	Incorrect	Error Rate (%)
1	ข (/k^h/)	K + 1	17	3	85.00	19	1	5
2	ค (/k^h/)	K + 2	16	4	80.00	16	4	20
3	ค (/k^h/)	K + 3	20	0	100.00	20	0	0
4	ถ (/t^h/)	T + 1	17	3	85.00	20	0	0
5	ฐ (/t^h/)	T + 2	15	5	75.00	20	0	0
6	ฒ (/t^h/)	T + 3	20	0	100.00	20	0	0
7	ท (/t^h/)	T + 4	20	0	100.00	20	0	0
8	ฏ (/$\bar{t}$/)	T + 5	17	3	85.00	20	0	0
9	ศ (/s/)	S + 1	11	9	55.00	16	4	20
10	ษ (/s/)	S + 2	15	5	75.00	18	2	10
11	ซ (/s/)	S + Q	15	5	75.00	20	0	0
12	ป (/p/)	P + 1	20	0	100.00	20	0	0
13	ผ (/p^h/)	P + 2	18	2	90.00	20	0	0
14	ภ (/p^h/)	P + 3	18	2	90.00	18	2	10
15	ฮ (/h/)	H + 1	18	2	90.00	20	0	0
16	ฎ (/d/)	D + 1	20	0	100.00	20	0	0
17	ฝ (/f/)	F + 1	18	2	90.00	20	0	0
18	ฬ (/l/)	L + 1	17	3	85.00	17	3	15
19	จ (/$t\widehat{\varepsilon}$/)	J1 + J2	20	0	100.00	20	0	0
20	ญ (/j/)	Y + 1	19	1	95.00	20	0	0
21	ณ (/n/)	N + 1	11	9	55.00	20	0	0
22	ง (/ŋ/)	N + G	12	8	60.00	20	0	5
23	ท (/t^h/)	T + H	17	3	85.00	20	0	0
24	ฉ (/$t\widehat{\varepsilon}^h$/)	C + H	19	1	95.00	20	0	0
Average accuracy					85.42	Average of error rate		3.54

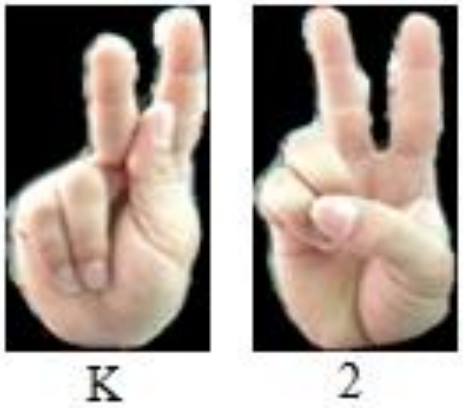
Table 6. The results of the three-stroke TFSL recognition system.

No.	Thai Letters	Rule	Classification			Stroke Detection		
			Correct	Incorrect	Accuracy (%)	Correct	Incorrect	Error Rate (%)
1	ธ (/t^h/)	T + H + 1	17	3	85.00	20	0	0
2	ช (/$t\widehat{\varepsilon}^h$/)	C + H + 1	13	7	65.00	19	1	5
3	ฌ (/$t\widehat{\varepsilon}^h$/)	C + H + 2	15	5	75.00	18	2	10
Average accuracy					75.00	Average of error rate		5

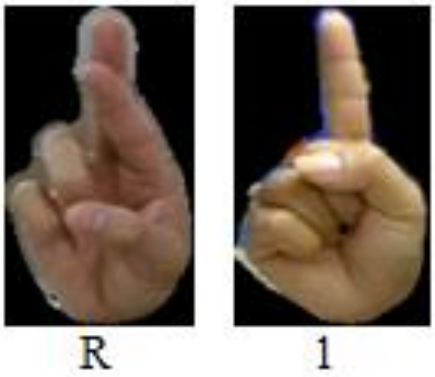
In terms of performance, the one-stroke TFSL recognition system shown in Table 4 had an average accuracy of 88.00%. According to the results, the system also encountered very similar sign language gesture discrimination problems in three groups: (a) a handful of gestures that use the thumb position ingress in different sign gestures for the letters T, S, M, N, and A, yielding lower accuracy for the letter N at 55%; (b) group gestures with two fingers, consisting of the index finger and middle finger, which are slightly different from the position of the thumb. This group consisted of letters K and 2 and lowered the accuracy of the letter K to 65%; and (c) a sign language gesture similar to raising a finger up one finger. The R gesture involves crossing between the index finger and ring finger, which is similar to the gesture for number 1, which involves holding up one index finger. This resulted in the accuracy of the letter R being 75%, as shown in Figure 20. These three groups of similarities among the sign language gestures resulted in a low accuracy average.



(**a**) T, S, N, M, and A gestures



(**b**) K and 2 gestures



(**c**) R and 1 gestures

Figure 20. A group of one-stroke TFSL signs with many similarities.

Table 5 shows the results of the TFSL two-stroke recognition system. The results of the experiment show that groups with very similar signs, such as T, S, M, N, and A, affected the accuracy rates in two-stroke sign language recognition systems, such as T + 2, S + 2, and S + Q providing an accuracy of 75%, N + G providing an accuracy of 60%, and S + 1 and N + 1 offering an accuracy of 55%. Sign language gesture number 2 was similar to K gestures. This affected the accuracy of K + 2, which had an accuracy rate of 80%. The similarity between sign language gestures affected the average accuracy of the two-stroke sign language recognition system, which was equal to 85.42%. In the sign language recognition system for two-stroke, we measured the error rate of stroke detection. The results show that sign language gestures with slight hand movements, such as K + 2 and S + 1, had a high error rate of 20%, affecting the overall accuracy of the system. The average error rate was 3.54%.

Three-stroke TFSL combines three sign language gestures to display one letter. Thus, three-stroke TFSL consists of three letters: T + H + 1, C + H + 1, and C + H + 2. The results showed that the accuracy of T + H + 1 was 85%, that of C + H + 2 was 75%, and that of

C + H + 1 was 65%, indication a low accuracy value. This low accuracy was due to several reasons, such as the detection of faulty strokes due to the use of multi-stroke sign language and sign language similarities. The average overall accuracy was 75%. The three-stroke has different posture movements, so there is a slight error rate, where the average error rate is 5%, as shown in Table 6.

Table 7 show that the multi-stroke TFSL recognition system has an overall average accuracy of 85.60%. The experiment used a training image for CNN modeling and testing with multi-stroke isolated TFSL. The results show that one-stroke tested 300 times was correct 264 times, incorrect 36 times, and had an average accuracy of 88%. For two-strokes, the average accuracy was 85.42% from 480 tests, with 410 correct and 70 incorrect. The use of three-strokes was tested 60 times and was found to be correct 45 times and incorrect 15 times with an average accuracy of 75%. Overall stroke detection of the system showed an average error rate of 3.70. The results show that sign language gestures with little movement affected stroke detection and caused misclassified.

Table 7. The overall results of the multi-stroke TFSL recognition system experiment.

No.	Stroke Group	Classification			Stroke Detection		
		Correct	Incorrect	Accuracy (%)	Correct	Incorrect	Error Rate (%)
1	One-stroke	264	36	88.00	-	-	-
2	Two-stroke	410	70	85.42	464	16	3.54
3	Three-stroke	45	15	75.00	57	3	5.00
Average accuracy				85.60	Average of error rate		3.70

5. Conclusions

The research focused on the development of a multi-stroke TFSL recognition system to support the use of complex background with deep learning. This study compared five CNN performance models. According to the results, the first and second structure formats are static CNN architecture, and the determined number of filters and size of the filters were the same for all layers, providing a minimal accuracy value. The third format uses an architectural style to determine the number of filters from ascending with the same size of the filter, which results in increased accuracy. The fourth format uses an ascending number of filters. The first layer uses large filters for learning the global feature, whereas the next layer uses smaller filters for learning the local feature. This results in higher accuracy. The fifth format is a mixed architectural style, designed based on AlexNet structure. Its first and second convolution layer increases the number of filters from ascending while the third and fourth layers show the number of a static filter rises from the second layer. However, the fifth and sixth layers use a fixed number of filters which drop from the two previous layers. For similar purpose, the large size of the filter is used in the first layer to learn global features, while the smaller filter is used in the next layer to learn the local feature. The results show that mixed architecture with global feature learning followed by local feature learning shows outperforming accuracy. By using the fifth format to train the system, the results of the overall study indicate that factors affecting the system's accuracy average include (1) very similar sign language gestures that negatively affect classification, resulting in lower accuracy average results; (2) low-movement sign language spelling gestures that affect the detection of multiple spelling gestures, causing faulty stroke detection; and (3) the spelling of multiple-stroke sign language, which affects the average accuracy since too much movement can sometimes lead to movement being detected between gestural changes, resulting in system recognition errors. A solution could be to improve the motion detection system to make the strokes of the sign language more accurate or to apply a long short-term memory network (LSTM) to enhance the recognition system's accuracy. In conclusion, the study results demonstrate that similarities in the gestures and strokes when finger-spelling Thai sign language caused decreases in accuracy. For future studies, further TFSL recognition system development is needed.

Author Contributions: Conceptualization, T.P. and P.S.; methodology, T.P.; validation, P.S.; investigation, T.P. and P.S.; data curation, T.P.; writing—original draft preparation, T.P. and P.S.; writing—review and editing, T.P. and P.S.; supervision, P.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the copyright of the Natural Language and Speech Processing Laboratory, Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen.

Acknowledgments: This study was funded by the Computer Science Department, Faculty of Science, Khon Kaen University, Khon Kaen, Thailand. The authors would like to extend our thanks to Sotsuksa Khon Kaen School for their assistance and support in the development of the Thai finger-spelling sign language data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, C.; Tian, Y.; Huenerfauth, M. Multi-Modality American Sign Language Recognition. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 2881–2885. [CrossRef]
2. Nakjai, P.; Katanyukul, T. Hand Sign Recognition for Thai Finger Spelling: An Application of Convolution Neural Network. *J. Signal Process. Syst.* **2019**, *91*, 131–146. [CrossRef]
3. Pariwat, T.; Seresangtakul, P. Thai Finger-Spelling Sign Language Recognition Employing PHOG and Local Features with KNN. *Int. J. Adv. Soft Comput. Appl.* **2019**, *11*, 94–107.
4. Pariwat, T.; Seresangtakul, P. Thai Finger-Spelling Sign Language Recognition Using Global and Local Features with SVM. In Proceedings of the 2017 9th International Conference on Knowledge and Smart Technology (KST), Chonburi, Thailand, 1–4 February 2017; pp. 116–120. [CrossRef]
5. Adhan, S.; Pintavirooj, C. Thai Sign Language Recognition by Using Geometric Invariant Feature and ANN Classification. In Proceedings of the 2016 9th Biomedical Engineering International Conference (BMEiCON), Laung Prabang, Laos, 7–9 December 2016; pp. 1–4. [CrossRef]
6. Saengsri, S.; Niennattrakul, V.; Ratanamahatana, C.A. TFRS: Thai Finger-Spelling Sign Language Recognition System. In Proceedings of the 2012 Second International Conference on Digital Information and Communication Technology and It's Applications (DICTAP), Bangkok, Thailand, 16–18 May 2012; pp. 457–462. [CrossRef]
7. Chansri, C.; Srinonchat, J. Reliability and Accuracy of Thai Sign Language Recognition with Kinect Sensor. In Proceedings of the 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Chiang Mai, Thailand, 28 June–1 July 2016; pp. 1–4. [CrossRef]
8. Silanon, K. Thai Finger-Spelling Recognition Using a Cascaded Classifier Based on Histogram of Orientation Gradient Features. *Comput. Intell. Neurosci.* **2017**, *2017*, 1–11. [CrossRef] [PubMed]
9. Tolentino, L.K.S.; Juan, R.O.S.; Thio-ac, A.C.; Pamahoy, M.A.B.; Forteza, J.R.R.; Garcia, X.J.O. Static Sign Language Recognition Using Deep Learning. *Int. J. Mach. Learn. Comput.* **2019**, *9*, 821–827. [CrossRef]
10. Vo, A.H.; Pham, V.-H.; Nguyen, B.T. Deep Learning for Vietnamese Sign Language Recognition in Video Sequence. *Int. J. Mach. Learn. Comput.* **2019**, *9*, 440–445. [CrossRef]
11. Goel, T.; Murugan, R. Deep Convolutional—Optimized Kernel Extreme Learning Machine Based Classifier for Face Recognition. *Comput. Electr. Eng.* **2020**, *85*, 106640. [CrossRef]
12. Wang, N.; Wang, Y.; Er, M.J. Review on Deep Learning Techniques for Marine Object Recognition: Architectures and Algorithms. *Control Eng. Pract.* **2020**, 104458. [CrossRef]
13. Cowton, J.; Kyriazakis, I.; Bacardit, J. Automated Individual Pig Localisation, Tracking and Behaviour Metric Extraction Using Deep Learning. *IEEE Access* **2019**, *7*, 108049–108060. [CrossRef]
14. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [CrossRef] [PubMed]
15. Rahim, M.A.; Islam, M.R.; Shin, J. Non-Touch Sign Word Recognition Based on Dynamic Hand Gesture Using Hybrid Segmentation and CNN Feature Fusion. *Appl. Sci.* **2019**, *9*, 3790. [CrossRef]
16. Koller, O.; Forster, J.; Ney, H. Continuous Sign Language Recognition: Towards Large Vocabulary Statistical Recognition Systems Handling Multiple Signers. *Comput. Vis. Image Underst.* **2015**, *141*, 108–125. [CrossRef]

17. Hamed, A.; Belal, N.A.; Mahar, K.M. Arabic Sign Language Alphabet Recognition Based on HOG-PCA Using Microsoft Kinect in Complex Backgrounds. In Proceedings of the 2016 IEEE 6th International Conference on Advanced Computing (IACC), Bhimavaram, India, 27–28 February 2016; pp. 451–458. [CrossRef]
18. Dahmani, D.; Larabi, S. User-Independent System for Sign Language Finger Spelling Recognition. *J. Vis. Commun. Image Represent.* **2014**, *25*, 1240–1250. [CrossRef]
19. Auephanwiriyakul, S.; Phitakwinai, S.; Suttapak, W.; Chanda, P.; Theera-Umpon, N. Thai Sign Language Translation Using Scale Invariant Feature Transform and Hidden Markov Models. *Pattern Recognit. Lett.* **2013**, *34*, 1291–1298. [CrossRef]
20. Yu, F.; Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. In Proceedings of the 2016 International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016.
21. Li, Y.; Zhang, X.; Chen, D. CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 1091–1100. [CrossRef]
22. Hosoe, H.; Sako, S.; Kwolek, B. Recognition of JSL Finger Spelling Using Convolutional Neural Networks. In Proceedings of the 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), Nagoya, Japan, 8–12 May 2017; pp. 85–88. [CrossRef]
23. Bunrit, S.; Inkian, T.; Kerdprasop, N.; Kerdprasop, K. Text-Independent Speaker Identification Using Deep Learning Model of Convolution Neural Network. *Int. J. Mach. Learn. Comput.* **2019**, *9*, 143–148. [CrossRef]
24. Colque, R.V.H.M.; Junior, C.A.C.; Schwartz, W.R. Histograms of Optical Flow Orientation and Magnitude to Detect Anomalous Events in Videos. In Proceedings of the 2015 28th SIBGRAPI Conference on Graphics, Patterns and Images, Salvador, Bahia, Brazil, 26–29 August 2015; pp. 126–133. [CrossRef]
25. Cheng, S.; Zhao, K.; Zhang, D. Abnormal Water Quality Monitoring Based on Visual Sensing of Three-Dimensional Motion Behavior of Fish. *Symmetry* **2019**, *11*, 1179. [CrossRef]
26. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]