Anatolian Journal of Computer Science
Volume:4 No:2 2019 © Anatolian Science
pp:54-67 ISSN:2548-1304

Smart Controlling Devices with Convolutional Neural Network

Zainab Shukur Mahmood Mahmood[1], Sait Ali Uymaz[2]*
*[1]*Konya Technical University, Department of Computer Engineering, Konya, Turkey (zainabdabagh8896@gmail.com)*
[2] Konya Technical University, Department of Computer Engineering, Konya, Turkey (sauymaz@ktun.edu.tr)
Received:Jan. 11, 2019
Accepted: Feb. 14, 2019
Published: Dec. 1, 2019

Abtract— The convolution neural network (CNN) consists of one or more Convolutional layers (usually in the sub-sampling step) and then one or more complete layers, as in the multiple layers of the neural network. The CNN structure is designed to transform the advantage of the structure into a two-dimensional input image (or another two-dimensional input, such as a speech signal). This result is obtained with local connections and are relevant to the weight, followed by the type of aggregation results in the translation of the static parts. Another advantage of the CNNs so that it is easier to train and they have a fewer number of parameters of the full network with the same number of hidden units. See all the tutorials on convolution and collecting for more details of this operation. The goal of this work is to design and execute a system using convolutional neural network that capable of automatically detect and recognize hand gesture through capture and process image in real time by using capture device (camera). The program is going to be on hand gesture detection with convolutional neural network, and it can be detect many different types of gestures at the same time that will be used for helping disabled, deaf or blind people that will helping them to control things in their house and helping them for fulfilling their demands. With this program, we can control electrical devices, mechanicals or electronics by using hands gestures. The program also can be used as teaching the deaf peoples to learn sign language. The code for training and creating the CNN model will be in Python using the Keras machine learning.

Keywords : *Ardunio, Sign Language , Convolutional Neural Network, Python.*

1. Introduction

Scientists and phantasy writers were seduced by the likelihood of building intelligent machines. The flexibility to know the visible world is unreal of such a machine. Consistent with a mathematician (Stockman ve Shapiro, 2001), one amongst the fathers of the fashionable information processing system and also the field of computing, believed that a computing machine computing information processing system would offer intelligence and also the ability to know a computer. Such goals have tried effective, and humanity continues to be incompatible with our engineer. The need to enhance the connection between humans and computers was a tool for brand new communication functions and new ways in which of handling machines (Shi ve ark., 2017). The quality language is usually used as an associate instrument once the image cannot be used or the writing method and the writing is troublesome, however the chance to possess the vision. In addition, language is that the most ordinarily notable and common approach for the listening hears (Rashed ve Hasan, 2017). The gestures area unit is so custom-made to the transmission of data that alternative strategies are not effective. In natural and easy, area unit may be used as a method or incorporated into multimodal communication as well as the word or author records. Specification strategies (Molchanov ve ark., 2015) may be developed for the advanced style of the interaction method to outline what forms of gestures area unit used, which implies they convey and what area unit the paradigms of the interaction. Humans use gestures in existence as a method of communication (for example, inform associate object to urge someone's attention on the topic, salutation a fan, requesting one thing by raising fingers). In gestural communication, the most effective example is language. American sign language (ASL) includes an associate English alphabet (TARRATACA, 2008). In other ways of hand gesture recognition have been defined on the basis of the fast wavelet transform they have learned through the fast wavelet transform (FWN) including the fuzzy decision support system (FDSS). Their interest in this research was to propose a new method for the FWN. FWN which has a hybrid structure (used as both wavelet and size measurement functions) provides hybrid weight carriers

when the image is rounded. The FWN classification stage was achieved by calculating the interstitial weights between experimental and training vectors. The latter consists of two types of transactions that are not within the same value range and which may affect computing distances. This can cause false confessions. In order to overcome this gap, a new classification strategy is proposed, which employs a human way of thinking that uses FDSS to calculate the similarities between test images and training. Discussion and comparison with other works where the results obtained show that the new position recognition tool works better than previously identified. (Bouchrika ve ark., 2014). When we see other studies (Oliveira ve ark., 2017), hand recognition is a very important area of computer vision. They have been working for years and didn't do a good enough exercise using ordinary camera inputs without sensors or many cameras. Computer interaction is largely based on new developments in CV and hand recognition is a very active area of research in this area. The Irish sign language (ISL) is known to be used by approximately 6,500 people with hearing impairment in Ireland. ISL does not support English or Ireland. It's a different language. In addition, ISL is estimated to be known by 50,000 non-deaf people (Liang ve Hu, 2015). This study presented a new data set for Irish sign language (ISL) and two comparisons were made. Data collection was collected by taking photographs of ISL forms and persons performing the movements. Then, import the frames from the videos. This produced a total of 52688 images of 23 common ISL forms. It then filters double images using a photo-replication process that identifies images that keep the dataset different. Use Principal Component Analysis (PCA) with k-nearest neighborhood (K-NN) and convolutional neural networks (CNN) for classification. They obtained an accuracy of 0.95. (Liang ve Hu, 2015)

2. Material and Methodology

2.1. Convolutional Neural Networks

Deep Learning (Li ve ark., 2015) and AI were the thrill words for 2016 by the tip of 2017, they need to become additional frequent and additionally confusing. Convolutional neural network or CNN foremost have a little deep dive into perceptrons. NN could be an assortment of many units/cells known as perceptrons that square measure binary linear classifiers as shown in figure 1.

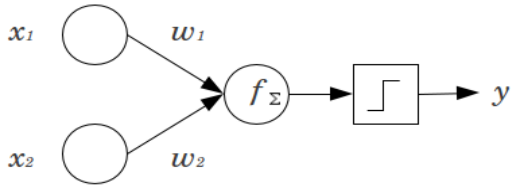

Figure 1. NN perceptrons

X_1 and X_2 are inputs are expanded with weights w_1 and w_2 and are collected along the port exploitation f and then $f = X_1 * w_1 + X_2 * w_2 + b$ (instead of the term bias). Currently, this procedure can be an alternative procedure, except for sensors that are usually an assembly. This performance is then evaluated by enabling the degree of partner that allows the selected category. Sine performance is the maximum normal activation performance used in binary classification. Now, if we accumulate a lot of input and link them to the function of using two cells stacked in another layer, this paper consists of two interfaces that are really related, the result of which is the input of the end (hidden layer). Use the f function and activation again to obtain another type of cell. This is the simplest of neural networks, as described in figure2. (Yadav)

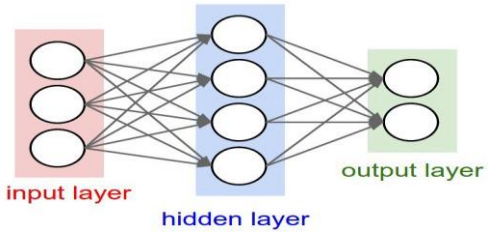

Figure 2. Convolution neural network layers.(Yadav)

In addition, NN are called Feed Forward Neural Network or FFNN, because the data flow is one-way, not periodic. At present, we all know the basics of cognition and FFNN, imagining many inputs associated with many hidden layers, such as a deep neural network or a familiar imaginary network known as the deep network. (Yadav)

The unit of the CNN-shaped bad processing unit has three classes of classes, in particular "Convolution", "Pooling" and "dense or fully connected". Our previous NN were a regular example of the "dense" NN layer since all classes were strictly interconnected.

Convolution Layer: 5x5 pixel image white color 1, black color 0, this image is reflected in 5X5 black and white image. Now imagine random 1s and 0s of a 3X3 matrix, and allow this matrix to try and run a matrix that is a subset of the image. (Yadav)

The task is to subtract the snippets of the image, so the abuse continues with the help of "optimization algorithms" to completely determine the values of 1s and 0s min in the 3X3 matrix. We tend to allow many of these filters to extract many options in a complete NN layer. A step of the 3X3 matrix is called "step".

A detailed read of a 3-channel (RGB) image manufacturing two convolution outputs victimization two 3-channel filters is provided, as described in figure 3.

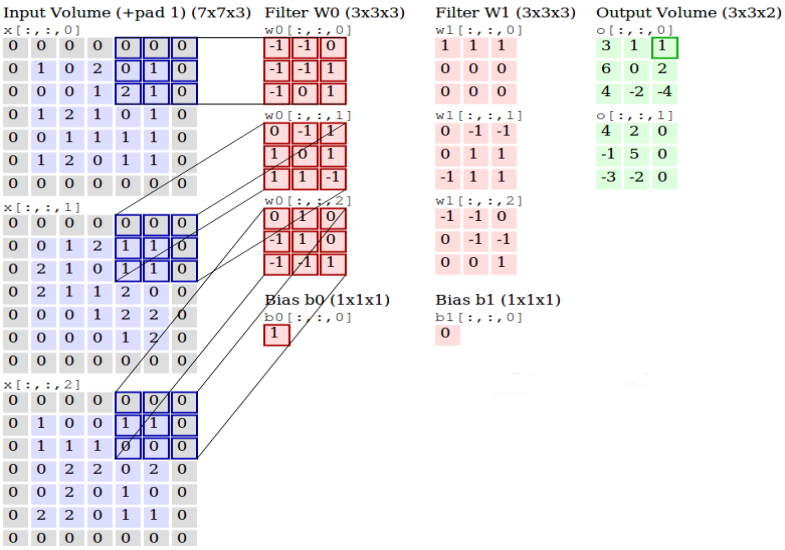

Figure 3. 3-Channel (RGB) image manufacturing (Yadav)

Those filters W0 and W1 are the “convolutions” and output is that the extracted function, a layer consisting of those filters might be a convolutional layer.

Pooling layer: This layer is used to reduce the dimensions of the victim's input to completely different functions. Typically, the Max Pool layer is generally used as a single flux layer. The build uses a 2x2 matrix and works the same as the embedding layer on the image, but this reduces the image itself. Below are two ways to group an image that uses "Max Pooling" or "Avg Pooling". (Yadav)

Dense Layer: This layer may be an absolutely connected layer between the activations and also the previous layer.

2.1.1. VGGNet

The competitor in the ILSVRC 2014 VGGNet contest is called by the community and developed by Simonyan and Zisserman. The VGGNet network consists of sixteen adaptive layers and is very attractive due to its highly unified design. Just like AlexNet, just 3x3 convolution, no matter how abundant the filter is. Training in four charge-processing units (GPUs) for 2-3 weeks. Right now, everyone in the community take the options out of the most popular alternate images. The composition of the VGGNet network load is open to the public and has been used in many alternative applications and difficulties as an abstraction of key features. However, VGGNet consists of 138 million parameters which may be a small amount difficult to use as described in figure 4. (Das, 2017)

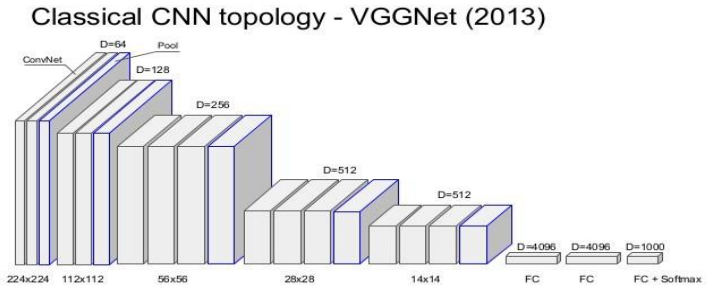

Figure 4. VGG NET (Das, 2017)

VGGNet-like architectures area unit characterised by

1. Using solely 3×3 convolutional layers stacked on high of every alternative in increasing depth.
2. Reducing volume size by soap pooling.
3. Fully connected layers at the tip of the network before a softmax classifier.

2.2 Filters

We applied many filters on the images within a highlighted region of interest (ROI) or the red rectangular. We used an OpenCV library to apply these filters. The reason for applying filters was to isolate the object from that background, also to increase the speed of process and prediction. We did not apply the filter on the whole image, we applied it just within ROI to increase the efficiency of the prediction and reduce the time that needed to predict. For coloration conversion, we tend to use the perform cv2.cvtColor(input_image, flag) wherever the flag determines the sort of conversion. For (BLUE, Green, Red) BGR grey conversion, we tend to use the flags cv2.COLOR we tend to modify the image among ROI from colored mode to grayscale mode. The second filter is cv2.GaussianBlur, as for one-dimensional signals, pictures can also be filtered with varied low-pass filters (LPF), high-pass filters (HPF), etc. An LPF helps in removing noise or blurring the image. An HPF filters help to find edges in a picture.(Anonymous14, 2013)

OpenCV provides operate, cv2.filter2D, to turn a kernel with a picture. As associate example, we are going to strive associate averaging filter on a picture. A 5x5 averaging filter kernel may be outlined as follows:

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

For each image element, a 5x5 window focuses on this image element, where all the pixels between this square are summarized and the result is divided into twenty-five. This is equal to the calculation of the values of common image elements in that window. This is done for all pixels in the image to provide the filtered image as described in figure 5.

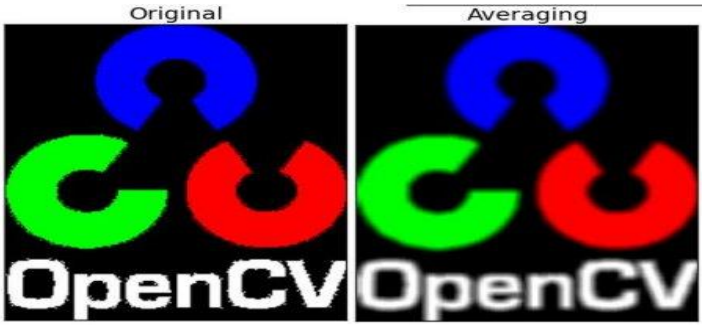

Figure 5. Applying smoothing filter to the image (Anonymous14, 2013)

A blurred image is obtained by combining the image with a low pass filter core. Useful for noise reduction. After this filter is applied, it deletes really high frequency content (eg noise, edges) from the image that causes the edges to blur. (Well, there are unclear techniques that do not blur the edges). OpenCV provides four main opacity techniques (average, athlete filtering, intermediate filtering, and binary filtering). In the math filter, the filter box consists of equal filter coefficients, the math core is used. The job is over, cv2.GaussianBlur(). We must always determine the size and length of the core that should be positive and bizarre. We tend to show the quality deviation in X, Y, Sigma X and Sigma Y modes in a versatile way. If sigma X is only nominal, Sigma Y will be considered a sufficient Sigma signal. If each is given as zero, it is calculated from the size of the kernel. Filtering mathematics is very effective for removing mathematical noise from the image. If you would like, you'll be able to produce a mathematicial kernel with the operate, cv2.getGaussianKernel. The on top of code will be changed for mathematician blurring:

blur = cv2.GaussianBlur(img,(5,5),0)

Here, if the value of the image element is greater than the threshold, a value (maybe white) is set, otherwise another value (possibly black) is set. User operator cv.threshold. The first argument is that the image provided must be a grayscale image. The second argument is the limit value used to classify the values of the image element. The third argument indicates whether the value of the image element (sometimes less) of the maxVal is the value of the edge. OpenCV provides completely different threshold formats and is determined by the fourth parameter of the study. Different types as described in figure 6.

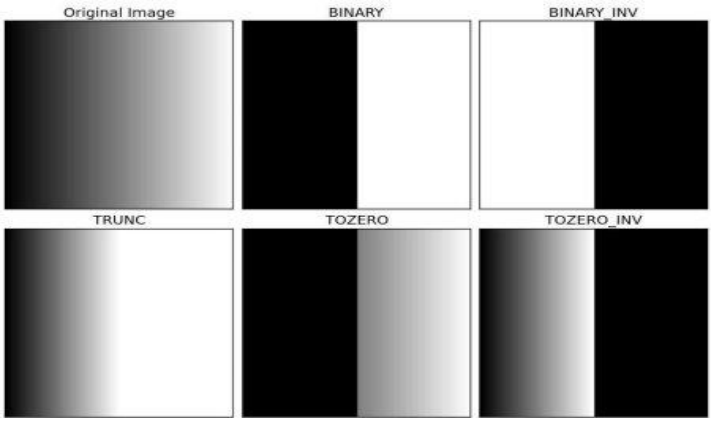

Figure 6. Different styles of thresholding (Anonymous16, 2017)

In the previous section, we tend to use a global value as a threshold value. However, situations where the picture has completely different lighting conditions in some areas should not be quite reasonable. In this case, we tend to choose the adaptive threshold. It calculates the edge algorithm for any low areas of the program image. Therefore, we tend to have very different thresholds for different areas of the same image, and also give higher results in the US for variable lighting images. Includes three custom input parameters and only one label.

Adaptive methodology - It decides however thresholding worth is calculated.

- cv.ADAPTIVE_THRESH_MEAN_C : threshold worth is that the mean of neighbourhood space.
- cv.ADAPTIVE_THRESH_GAUSSIAN_C : threshold worth is that the weighted add of neighbourhood values wherever weights are a mathematicial indow.

Block Size - It decides the scale of neighbourhood space.

C - it's simply a relentless that is ablated from the mean or weighted mean calculated as described in figure 7. (Anonymous16, 2017)

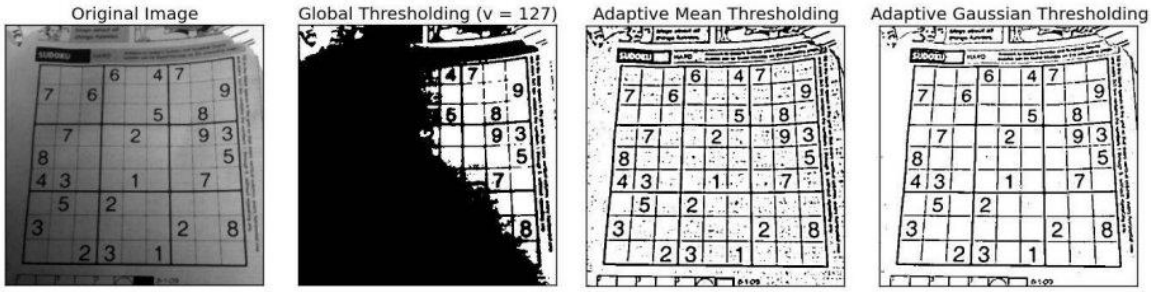

Figure 7. Apply different type of adaptive threshold image (Anonymous16, 2017)

For Otsu binary encoding, as a retVal parameter. Its use comes as soon as Otsu Binarization is selected. As a global threshold, they tend to use a nursing assistant to change the threshold value. It is easy to calculate the threshold value of a bar graph image of a binary image. (For non-double-sided images, encryption will not be true.) For this, cv.threshold is executed by passing an extra tag named cv.THRESH_OTSU. For the threshold value, simply go to zero. The formula then finds the most appropriate value and returns you because of retVal, which is the second output. If the Otsu threshold is not used, retVal will be the same amount that you deserve. Check the following example. Image input can be an image of a note. In the first case, you applied the world threshold at price 127. In the second case, you applied the OTSȩ threshold directly. In the third case, the 5x5 math kernel was used to filter noise, then applied the grass threshold as described in figure 8 (Anonymous16, 2017).

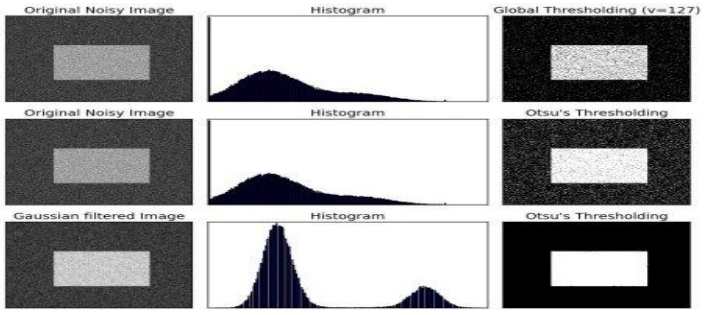

Figure 8. Otsu's Binsization (Anonymous16, 2017)

3. The Proposed Method

The goal of this thesis is to Design and execute a system using convolutional neural network (CNN) that capable of automatically detect and recognize hand gesture through capture and process image in real time by using capture device (camera). The code for training and creating the CNN model will be in Python and the dataset of each gesture has two folders, test and training dataset of data. Data has been taken within the application within the region of interest and recording pictures. For training (Images feature) approximately 1500 images of each case has been recorded with an additional approximately 600 images for validation (test images) as described in figure 9.

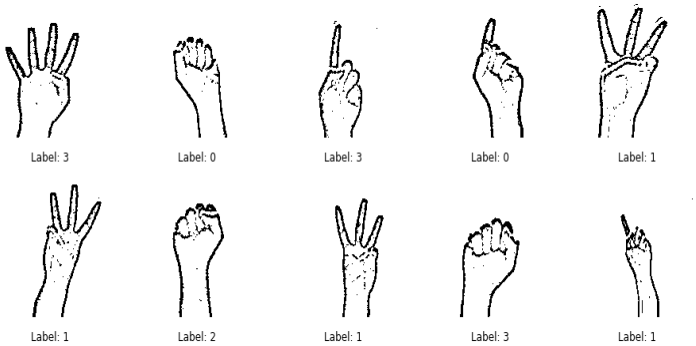

Figure 9. Gesture's images with their labels

We wrote a custom hand detecting algorithm. The system will start from capture device (camera) as input unit and will capture the specific gesture and handle it through processing unit (Python), then send the specific sign that carry the specific function to esp8266 through third-party (Firebase database hosting) to control specific device. The code for training and creating the CNN model was created in Python using the Keras helper function for machine learning library. In addition, Python run client side socket that waiting for image recognition requests from capture device (camera) and send the functions to output unit (Arduino) through the internet. There is a 3rd party environment (firebase), works as medium to connect the hardware with the software. With python, we have the ability to learn the system to new gestures.

If the captured gesture was not defined by the database the client socket will be as busy for detect another gestures. In other case if the gesture is defined, the client socket recognize the gesture that had trained earlier in the system and send the result to host server (Firebase) to do the rest. The server side after receive the result from client side, the Arduino will analyze the result and send the outputs as (ON/OFF) based on result of the database. The flow chart of the system is illustrated below in figure 10.

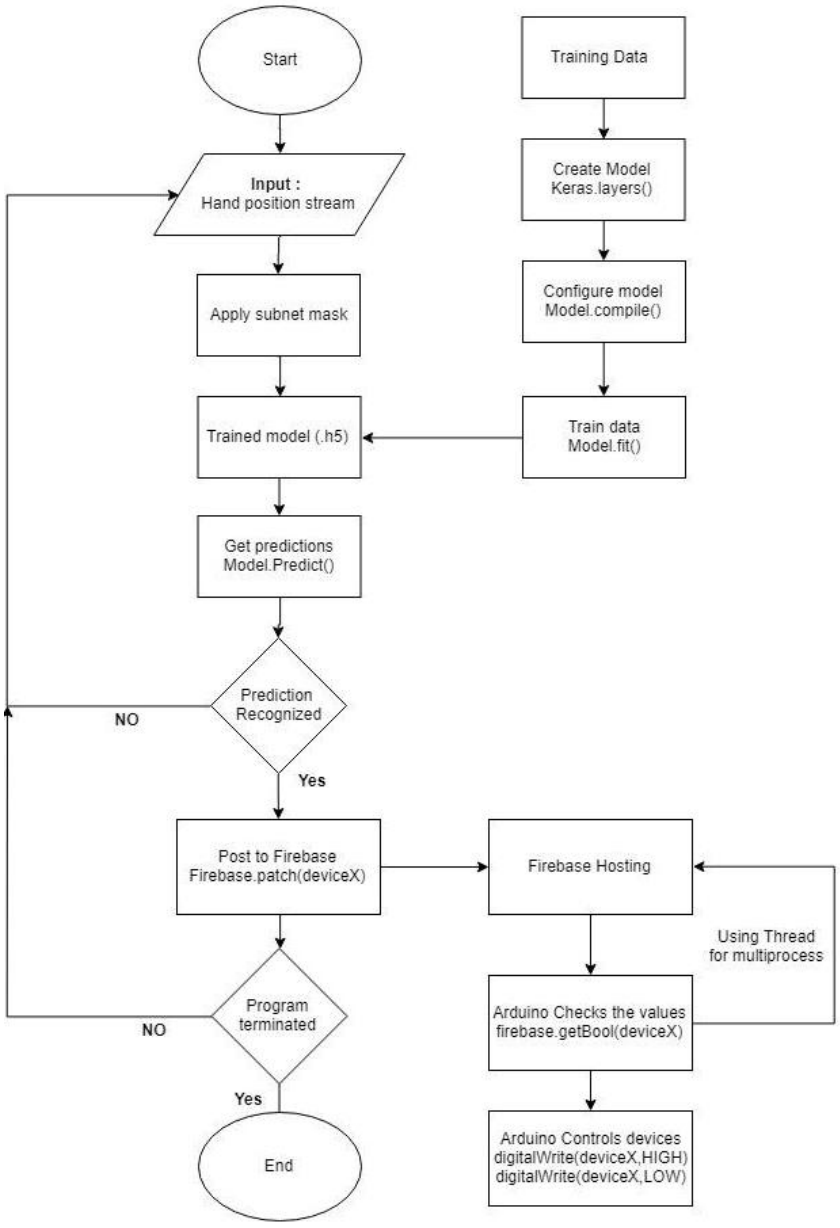

Figure 10. Flowchart of the system

4. Experiments

4.1 CNN system

In deep learning, a convolutional neural network (CNN, or ConvNet) could be a category of deep neural networks, most typically applied to analyzing visual representational process. CNNs use a variation of multilayer perceptrons designed to want smallest preprocessing. They are conjointly referred to as shift invariant rspace invariant artificial neural networks (SIANN), supported their shared-weights design and translation unchangingness characteristics. The data seems reasonable so far, we will define the CNN model (figure 4.1) for training and use it in our application and the model illustrated below is not fully enhanced but it seems to work well for our purposes. We used 12 layer. The summery result that we obtained from the created model of the neural network is illustrated below in figure 11.

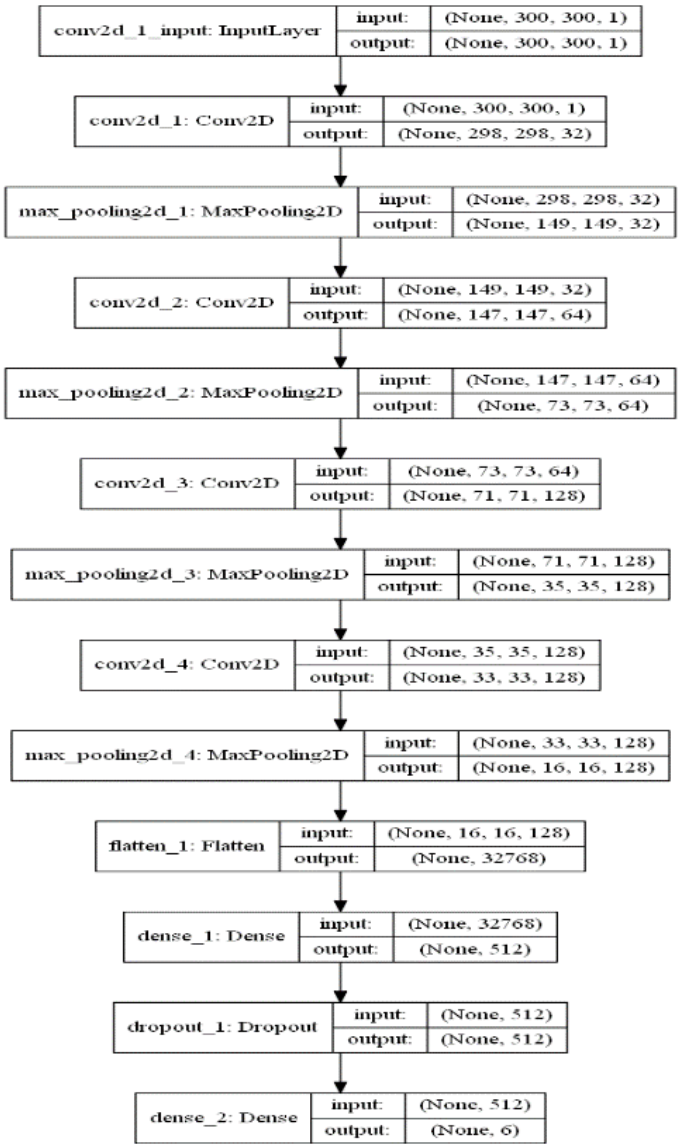

Figure 11. Model of CNN

4.2 Dataset

The data is collected from the program. We used Keras's ImageDataGenerator and not only label of the data from the directories, but also to increase data slightly with transformation, rotations, zooms and reflection. The reflection will help to ensure that the data are not biased to a particular hand. To ensure that the model is not bias

towards any particular label, we must have almost equal examples of each sample as described in figure 12. To define a dataset for 1500 images we give a name (dataset1) and for 600 images (dataset2).

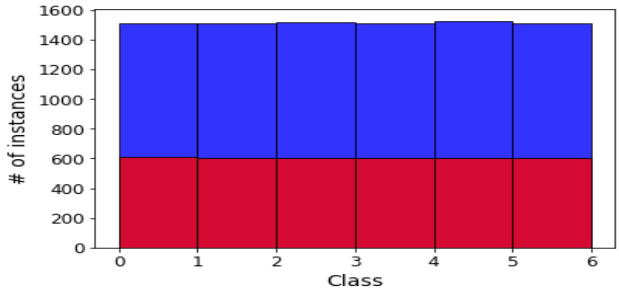

Figure 12. The number of examples were selected from instances for training

After that, we'll do some quick checks to make sure our data is properly categorize

Where : 128 : number of nbatch, 300,300 : input image 300 x 300 pixel, 1: number of channel (128, 6) 128 number of nbatch for 6 classes as described in figure 13.

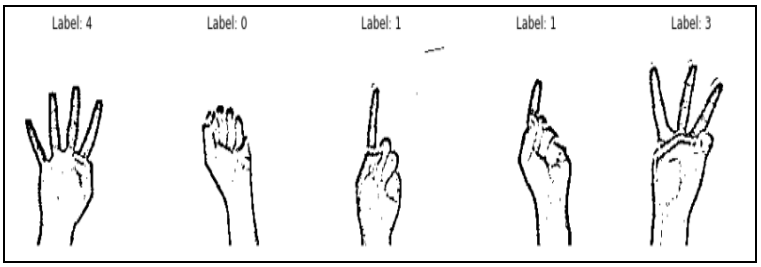

Figure 13. Illustrate that the data had been labeled correctly

4.3 Training

We used Keras function to train our model and save it to file with extension (.h5). We are not doing any hyper-parameter optimization so did not define a separate testing and validation set. We could evaluate out model in real time within out program. The result and the accuracy for 71 epoch that we got from the training is 99.8%. We applied the graphing helper function matplotlib to plot the loos and the accuracy of the training operation as described in figure 14.

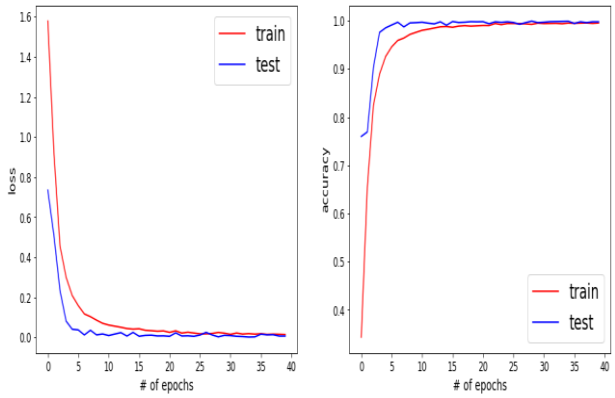

Figure 14. Illustrates the loss and accuracy for training operation

From the above figure it seems that we're able to achieve near perfect (> 99%) accuracy.

4.4 Performance and Error Analysis

Out model is working very good but we expanded the testing set and we applied some augmentation to check the cases were our model failed to predict. After that we calculated the samples were the predict was not equal to dataset and we found its equal to 131 examples from 3632 images that belonging to six classes. Then we displayed the examples which predictions were not equal to samples as described in figure 15.

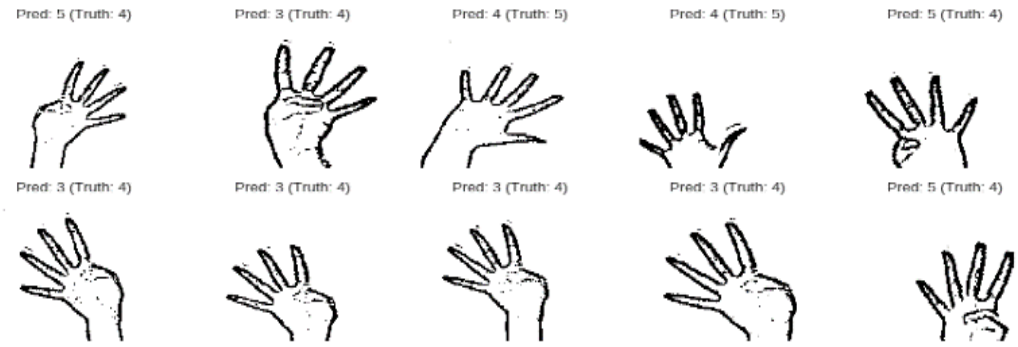

Figure 15. Samples which failed to predict

There are some of examples above where the fingers are out of the frame pf the process during reinforcement and predict its defective too. These problems might be present during training and it is difficult to neglect the model to obtain false expectations in these bad situations.

The network is work very well overall, but has the biggest problem with counting four fingers when the four figure are completely rotate to the side and the system usually confusing three fingers. This type of network is not constant rotation (Anonymous12, 2016) the issue of malfunction prediction, it is can be improved with more data or modified the model. For the purposes of this project, this issue is considered acceptable but can be improved in the future.

We used confusion matrix which helped us to summarize the cases that are misclassified as shown blow in figure 16 and Table-1: With dataset2 performance

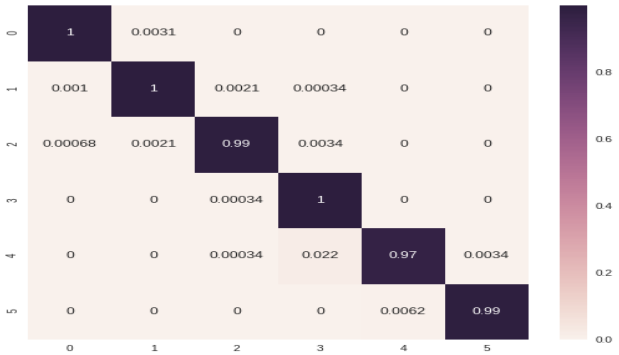

Figure 16. To summarize the cases that are misclassified with confusion matrix

Table 1. Dataset2 performance

	Accuracy	Precision	Sensitivity	Specificity	F-measuer
Class N	99.92%	99.83%	99.69%	99.97%	99.76%
Class L	99.86%	99.48%	99.66%	99.90%	99.57%
Class R	99.85%	99.72%	99.38%	99.44%	99.55%
Class A	99.56%	99.49%	99.97%	99.48%	99.71%
Class P	99.47%	99.36%	97.41%	99.88%	98.38%
Class V	99.84%	99.66%	99.38%	99.93%	99.52%
Aver of all classes	99.75%	99.25%	99.25%	99.25%	99.25%

In the first time, we used dataset1 to the images in order to get the best ones and eliminate the bad ones. Now, we will train the Model using dataset1 process to check the prediction and loss result and the number of images were ~1500 images (instances) per classes as shown in figure 17 below.

Figure 17. Number of instances were token

For training, we used the graphic card (NVidia GeForce 940MX). The we installed CUDA and CUdnn libraries that utilized to use the (GPU) with python. Then we write some lines of code to check whether the system fixed the GPU correctly in the system or not and as a final result, the tensorflow could correctly deal with GPU of the graphic card.

The CPU that used for train was (core i5 7th Gen – 7200U @2.50GHz (4 CPUs,~2.7GHz)) and GPU for graphic card was GeForce 940MX, we checked the elapsed time and rate speed between the CPU and GPU to train the model for all instances (~1500 per classes). We could find that the time that needs the system to train the model with CPU was 3:50 (Hour/Epoch) and for 40 epoch, we need ~160 hour to train the model. After that we checked the time that need to train the model with GPU We could find that the system needs 30-40 minutes/epoch and we needed just ~20 hours to train our model.

The model of the system and the result that we gained from the training with GPU as shown in figure 18.

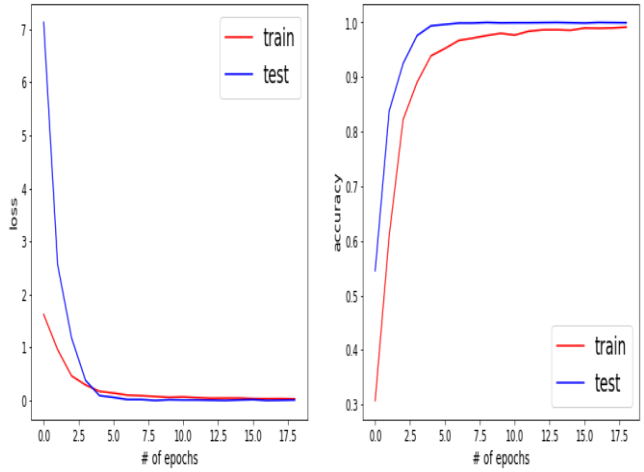

Figure 18. loss and accuracy dataset1 for both test and train instances.

Since there is dropout being applied in the network and additional (random) augmentation being applied to the training set it is not surprising that the validation set is out-performing the training set. Nonetheless, it seems that in either case we're able to achieve near perfect (> 99%) accuracy.

Our model is performing so well that there are not many cases where the model has failed for us to understand our model better. When we increased the number of instances we could improve the system and the system had able to understand the instances better than before, and the system reduced the number of bad instances to 62 examples. Not like before where the number of examples were 131 examples of bad predictions. The reason of the reduction because we increased the number of instances so the system could understand the model better as shown in Figure 19, Figure 20 and Table 2: dataset1 performance.

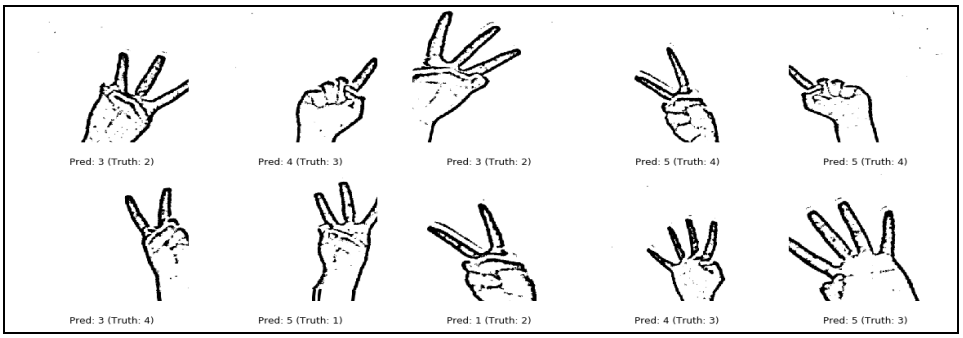

Figure 19. Some of bad examples for dataset1 process

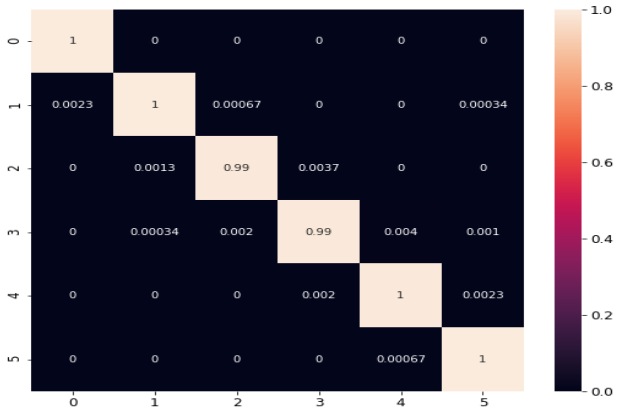

Figure 20. Confusion matrix for dataset1 process

Table 2. dataset1 performance

	Accuracy	Precision	Sensitivity	Specificity	F-measure
Class N	99.96%	99.77%	100%	99.95%	99.89%
Class L	99.92%	99.48%	99.67%	99.97%	99.75%
Class R	99.87%	99.73%	99.50%	99.95%	99.61%
Class A	99.78%	99.43%	99.26%	99.89%	99.35%
Class p	99.85%	99.545	99.57%	99.91%	99.55%
Class v	99.93%	99.64%	99.93%	99.93%	99.78%
Aver of all classes	94.89%	99.66%	85.42%	99.66%	91.995

As we mentioned before the confusion matrix helps summarize the cases that are misclassified as shown in figure 20. In general we saw before that the networks seems to usually under-count the number of fingers and this is probably at least in part to the issues in augmentation that we noted previously. In addition, as we noted before the network performs quite well in general but has the most trouble counting four fingers, usually confusing it for three fingers. This issue seems most prevalent when the four fingers are quite rotated to the side. As the convolutional network is not rotation invariant, but by increasing the number of instances, we improved the model with more data and the model could passes quite bit this issue. For the purposes of this project, this issue is deemed acceptable but could be improved in the future.

We have compared the values of accuracy, loss and examples of bad prediction in both cases, using dataset2 and dataset1 as shown in table 3 below:

Table 3. Compared between dataset1 and non-dataset1 cases.

	dataset1	Dataset2
Accuracy	99.65%	99.25%
Val- Loss	0.9%	0.68%
Number of bad Prediction	62	131

The reason of decreased the number of instances was to reduce the time that need to train the model and get the best prediction with minimum number of instances (images). Shown in Tabel-4: Compared between own results and with the literature.

Table 4. Compared between own results and with the literature.

Method	Recognized Gestures	Total Gestures used For Training And Testing	Recognition Percentage	Database used
(Wysoski ve ark., 2002)	26	1040	DP 98.8% MLP 98.7%	American Sign Language (ASL)
(Hasan ve Misra, 2011)	6	60	normal method 84% Scaling dataset1 method 95%	Their Database
(Kulkarni ve Lokhande, 2010)	26	208	92.78%	American Sign Language (ASL)
(Zhao ve ark., 2008)	0-9 numbers	298 video sequence for isolated gestures/ 270 video sequence for continuous gestures	90.45%	Recognize Arabic numbers from 0 to 9.
(Biswas, 2018)	5 static/ 12 dynamic gestures	Totally 240 data are trained and then the trained are tested	98.3%	5 static gestures and 12 dynamic gestures.
(Stergiopoulou ve Papamarkos, 2009)	31	130 for testing	90.45%	Their Database
(Hasan ve Mishra, 2012a)	6	60	100% for more than 4 gestures	Their Database
(Hasan ve Mishra, 2012b)	20	200	100% for 14 gestures, and >90 for 15-20 gestures	Their Database
Own work	6	3632	With dataset1 99.65%	Own Database
	6	9081	With dataset2 99.25%	

5. Conclusion

Many studies have been conducted in the world about sign language recognition and are still ongoing. When the studies about hand gesture recognition were evaluated, many problems were encountered. In our project, we tend to created management devices for hand gesture recognition by convolutional neural network that have ability to find and acknowledge hand gestures through captures the image via camera and method in real time. With this method, it's the power to detects many alternative styles of gestures at same time which will be used for serving to disabled, deaf or blind those who can serving to them to regulate things in their house and serving to them for fulfilling their demands. With this this program, dominant electrical devises, mechanicals or electronics are easier simply with hands gestures. The code for coaching and making CNN model are in Python by using the Keras, Theano that use for machine learning and openCV. We have got with success enforced this project and are ready to attain satisfactory results. In future studies, a comprehensive study can be developed, which can be used on the mobile platform, where the designer system will remove restrictions on environmental conditions, not only on hands, but also on the face and lips.

6. References

Anonymous12, 2016, About CNN, kernels and scale/rotation invariance, https://stats.stackexchange.com/questions/239076/about-cnn-kernels-and-scale-ro, [Accessed:14 Dec 2018].

Anonymous14, 2013, Changing Color-space, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html, [Accessed:31 Dec 2018].

Anonymous16, 2017, Image Thresholding, https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html, [Accessed:31 Dec 2017].

Biswas, A., 2018, Finger Detection for Hand Gesture Recognition Using Circular Hough Transform, In: Advances in Communication, Devices and Networking, Eds: Springer, p. 651-660.

Bouchrika, T., Jemai, O., Zaied, M. ve Amar, C. B., 2014, A new hand posture recognizer based on hybrid wavelet network including a fuzzy decision support system, *International Conference on Intelligent Data Engineering and Automated Learning*, 183-190.

Das, S., 2017, CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more …. [Online], https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5, [Accessed:12 Dec 2018].

Hasan, M. M. ve Misra, P. K., 2011, Brightness factor matching for gesture recognition system using scaled dataset1, *International Journal of Computer Science & Information Technology*, 3 (2).

Hasan, M. M. ve Mishra, P. K., 2012a, Features fitting using multivariate gaussian distribution for hand gesture recognition, *International Journal of Computer Science & Emerging Technologies IJCSET*, 3 (2), 73-80.

Hasan, M. M. ve Mishra, P. K., 2012b, Robust gesture recognition using gaussian distribution for features fitting, *International Journal of Machine Learning and Computing*, 2 (3), 266.

Kulkarni, V. S. ve Lokhande, S., 2010, Appearance based recognition of american sign language using gesture segmentation, *International Journal on Computer Science and Engineering*, 2 (03), 560-565.

Li, H., Lin, Z., Shen, X., Brandt, J. ve Hua, G., 2015, A convolutional neural network cascade for face detection, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5325-5334.

Liang, M. ve Hu, X., 2015, Recurrent convolutional neural network for object recognition, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3367-3375.

Molchanov, P., Gupta, S., Kim, K. ve Kautz, J., 2015, Hand gesture recognition with 3D convolutional neural networks, *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 1-7.

Oliveira, M., Chatbri, H., Little, S., Ferstl, Y., O'Connor, N. E. ve Sutherland, A., 2017, Irish sign language recognition using principal component analysis and convolutional neural networks, *Digital Image Computing: Techniques and Applications (DICTA), 2017 International Conference on*, 1-8.

Rashed, J. R. ve Hasan, H. A., 2017, NEW METHOD FOR HAND GESTURE RECOGNITION USING WAVELET NEURAL NETWORK, *Journal of Engineering and Sustainable Development*, 21 (1), 65-73.

Shi, B., Bai, X. ve Yao, C., 2017, An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39 (11), 2298-2304.

Stergiopoulou, E. ve Papamarkos, N., 2009, Hand gesture recognition using a neural network shape fitting technique, *Engineering Applications of Artificial Intelligence*, 22 (8), 1141-1158.

Stockman, G. ve Shapiro, L. G., 2001, Computer Vision. 1st, Upper Saddle River, NJ, USA: Prentice Hall PTR.

TARRATACA, L., 2008, A gesture recognition System using smartphones, *Dissertação (Mestrado em Sistemas de Informação e Engenharia da Computação …*.

Wysoski, S. G., Lamar, M. V., Kuroyanagi, S. ve Iwata, A., 2002, A rotation invariant approach on static-gesture recognition using boundary histograms and neural networks, *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, 2137-2141.

Yadav, P., A deeper understanding of NNets (Part 1)—CNNs [online], https://towardsdatascience.com/a-deeper-understanding-of-nnets-part-1-cnns-263a6e3ac61?fbclid=IwAR1-GhBSZYgGFYfDPN9F02T9eZwowUgRRDnA6p8KOypO73ID68op5de8h0Q, [Accessed:12 Dec 2018].

Zhao, S., Tan, W., Wen, S. ve Liu, Y., 2008, An improved algorithm of hand gesture recognition under intricate background, *International Conference on Intelligent Robotics and Applications*, 786-794.