Research Article
Journal of Intelligent Systems: Theory and Applications 4(1) (2021) 17-23
DOI: 10.38016/jista.808458

Recognition of Sign Language Letters Using Image Processing and Deep Learning Methods

Ali Öztürk [1*], Melih Karatekin [2], İsa Alperen Saylar [3], Nazım Bahadır Bardakcı[4]

[1,2,3,4] KTO Karatay University, Faculty of Engineering, Department of Computer Engineering, KONYA/Turkey

aliozturk2002@gmail.com, sefamelihkaratekin@gmail.com, alpisa1907@gmail.com, nazimbahadirbardakci@gmail.com

Abstract
In order for people to be able to communicate with each other, they must be able to agree mutually. Communication is quite difficult for individuals with hearing problems. Such individuals make their lives much more difficult by isolating themselves from society. The people living with hearing loss can understand the contact person with often lip-reading method, but it is quite difficult for them to express themselves to the people. Since the use of sign language has not become widespread around the world, the number of people who know sign language is very low, except for individuals with hearing disabilities. In this study, it was achieved to dynamically recognize the movements of the sign language finger alphabet via image processing using deep learning methods and to translate it into writing. Accordingly, it is aimed to facilitate communication between people who do not know the sign language in daily life and people with hearing loss. The input given to the system is an image of the hand showing any letter from the alphabet. The image of the hand is interpreted by deep learning methods in the system, and it is compared to one of the letters in the alphabet and an output with the similarity ratio to this letter is displayed on the screen. The system has been tested with a total of 1300 images. The overall accuracy rate of the system was calculated as 88% where true positive rate was 87% and false negative rate was 13%.

Keywords: Sign Language, Image Processing, Deep Learning, Image Analysis, Object Detection.

Görüntü İşleme ve Derin Öğrenme Yöntemleri Kullanılarak İşaret Dili Harflerinin Tanınması

Öz

İnsanların birbirleriyle iletişim kurabilmeleri için karşılıklı olarak anlaşabilmesi gerekmektedir. İşitme problemi yaşayan bireyler için iletişim kurmak oldukça zordur. Bu tür bireyler kendilerini toplumdan soyutlayarak yaşamlarını çok daha zor hale getirmektedir. İşitme kaybı yaşayan insanlar iletişimde bulunurken karşısındaki kişiyi genellikle dudak okuma yöntemi ile anlayabilmektedir fakat karşılarındaki insanlara kendilerini ifade etmekte oldukça zorlanmaktadır. İşaret dili kullanımının dünya genelinde yeterince yaygınlaşmamış olması sebebiyle, işitme engelli bireyler dışında işaret dilini bilen kişi sayısı oldukça azdır. Bu çalışmada, işaret dili parmak alfabesine ait hareketlerin, derin öğrenme yöntemleri kullanılarak görüntü işleme ile tanınması ve yazıya çevrilmesi dinamik olarak sağlanmıştır. Bu doğrultuda, gündelik yaşamda işaret dilini bilmeyen insanlar ile işitme kaybı olan insanlar arasındaki iletişimi kolaylaştırmak hedeflenmiştir. Sisteme verilen girdi, alfabeden herhangi bir harfi gösteren elin görüntüsüdür. Elin görüntüsü, sistemde derin öğrenme yöntemleriyle yorumlanarak, alfabedeki harflerden biriyle eşleştirilmekte ve bu harfe benzerlik oranı ile birlikte ekrana yazdırılmaktadır. Geliştirilen sistem, 1300 resim ile test edilmiştir. Sistemin doğruluk oranı %88, gerçek pozitif oranı %87 ve yanlış negatif oranı %17 olarak bulunmuştur.

Anahtar kelimeler: İşaret Dili, Görüntü İşleme, Derin Öğrenme, Görüntü Analizi, Nesne Tespiti

1. Introduction

Creatures use the signals transmitted through various channels when communicating. These communication channels are audible, visual, chemical etc. Sign language; is a silent, visual language created by the handicapped and speech impaired people by using hand gestures, facial expressions and body language as a whole to communicate with each other. This non-universal language varies from country to

* Corresponding Author.
E-mail: aliozturk2002@gmail.com

Recieved :09 Oct. 2020
Revision :22 Dec. 2020
Accepted :22 Dec. 2020

country. In this sense, there are more than 200 sign languages in the world. American, British, German, French, Italian, Indian and Turkish sign languages are some of them. In this article, it is aimed to recognize the Turkish sign language and some English sign language letters (Q, W, X) with image processing using deep learning methods and translate it into writing. Many studies have been done on this subject. As a result of our researches, approximately 4000 image dataset which were specifically created for the Turkish Sign Language alphabet were selected. From this dataset, approximately 1600 photographs were edited and tagged and used in the system training phase. The system was developed with the Python programming language. TensorFlow and OpenCV libraries were used as image processing tools. Image Labeling and Object Detection methods were applied. Thus, the labeling, training and testing of the dataset were achieved. Faster-R-CNN (Convolutional Neural Network) was determined as the deep learning model. In addition, the accuracy rates and performance analysis of the obtained results were performed. As the output of the study, a dynamic and high performance system was developed to facilitate communication between people who do not know sign language and people with hearing impairment.

1.1. Literature Survey

In their paper, Ashish and Aarti (2016) proposed a system hand gesture recognition system where image color analysis and image segmentation methods are used. The proposed system prepared by Sandhya and Ananya (2018) aimed to recognize American Sign Language and convert it into text. The input given to the system was a hand image showing the required alphabet. The histogram of the input image was calculated and checked for similarity to the histograms of previously saved images using the Bhattacharyya Distance Metric. OpenCV was used in the proposed system. The CNN, Machine Learning (ML) methods and depth-sensing technology were used in the study which was performed by Gaikwad et al. (2019). They used a pre-trained GoogleNet architecture. In the study of Tamura and Kawasaki (1988), a 3D dictionary was created from hand shapes. According to this dictionary, 2-dimensional images were interpreted and classified according to their features.

Kumarage et al. (2011) proposed to subdivide the transactions for recognition via parallel processing and mapping the motion data to static data representations, Also, the issue of matching sign language gestures with linear / nonlinear equations was mentioned. In the paper which emerged as a result of the research of Tan et al. (2020), a CNN with spatial pyramid pooling for vision-based hand gesture recognition was introduced. The performance of the proposed method was evaluated on American sign language datasets and hand gesture dataset. Pramada et al (2013) designed an intelligent system using the concepts of image processing, machine learning and artificial intelligence where sign language alphabet was digitally encoded by filtering. In their research Shivashankara and Srinath (2018), proposed a framework where the 24 static alphabets of the American Sign Language (letter J and letter Z not included) were translated into English text. They achieved an average recognition rate of 98.21%. In their study, Vargas et al (2011) presented an image pattern recognition system using neural network for the identification of sign language. The system had several stored images that showed the specific symbol in a kind of language which was employed to train a multilayer neural network using a backpropagation algorithm. They enabled the definition of the sign language by using image processing with various filtering methods and algorithms. They also stated that different features can be added to improve the accuracy of the system.

In the work of Unutmaz et al. (2019), the Turkish sign language recognition was investigated by hand gestures made in front of Kinect device and converted into writing. Convolutional Artificial Networks (CNN), Support Vector Machines (SVM), k-Nearest Neighbours (kNN) and Decision Trees (DT) were used for recognition and their performances were compared. When the manually extracted video frames were used via data augmentation, the accuracy rate reached to %99. However, when the sequential video frames were used to imitate real-time recognition behaviour, the maximum accuracy rate decreased to %81.

Çelik and Odabaş (2020) suggested a system that automatically detects Turkish sign language for 29 letters in the Turkish alphabet and 10 numbers by hand gestures made in front of the camera. This method has been developed with CNN modeling for sign detection using frames of videos. Different success rates have been achieved by manually processing the frames of videos. It has been reached a 53.8% accuracy rate without preprocessing of hand images using only CNN. By means of manual hand image extraction, %91 accuracy rate has been achieved. When more than one CNNs have been used with LSTM to manually capture the hand movements in the video frames, the accuracy rate has increased to 97%.

2. Materials and Methods

Python 3.7 was preferred as programming language in this project for image processing. Among the deep learning libraries, TensorFlow was chosen as the most suitable one for our system. The training and determination of the dataset were carried out via the TensorFlow library. The video card was one of the main parts of the overall system. The OpenCV libraries on the other hand, enabled the camera to be controlled,

the data to be tagged and the images to be filtered. In addition, it showed which class the detected object belongs to in a rectangle, with a percentage of accuracy.

Any of the drivers prepared by the video card manufacturers for deep learning was required to be able to train the system. These drivers have been developed to make training more efficient, fast and successful. The equipment which was used during training was the Nvidia Ge-Force GTX 1060 graphics card with a capacity of 3 GB, and the CUDA core was 1280.

To apply deep learning methods on this video card, CUDA and cuDNN drivers were downloaded from the video card manufacturer's official page. The most suitable versions of the drivers for the system are CUDA 8.0 and cuDNN 7.1.4. After installing the drivers and assigning their paths to the system variables, a virtual environment was created on the Anaconda IDE. Since the system had a complex structure in itself, it was important to download all libraries completely to the virtual environment.

The algorithm model to be used in training has been determined as Faster R-CNN. As a result of our research, we observed that the most appropriate algorithm model in terms of hardware and object classification time was Faster-R-CNN. Faster R-CNN applies CNN method on to the images. In CNN, a mask filter navigates on the image, the filter converts each mask into a pixel and calculates on the point where it is located, then stores the results in the output matrix. This last reached output matrix is called "feature map". Feature Map makes predictions based on image tagging. Faster RCNN architecture is a combination of region proposal network (RPN) which proposes regions and Fast RCNN detector which uses these proposed regions to give final bounding boxes of the object. An RPN takes feature map as input and outputs a set of rectangular object proposals, each with an objectness score. The objectness measures score between object and background. Faster RCNN uses anchor boxes of 3 aspect ratios and 3 scales. Thus for each pixel in the feature map, there are 9 anchor boxes. The region proposal time with selective search is 2 sec per image, whereas with RPN it's just 10ms (Shaoqing et al., 2017).

3. The Proposed System

The main steps of the proposed system is shown in Figure 1.

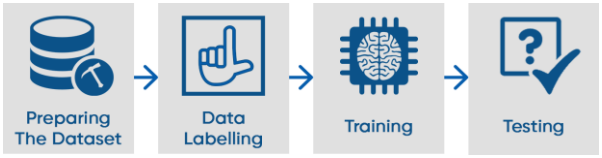

Figure 1. The main steps of the proposed system

3.1. Preparing The Dataset

In this step, dataset research was carried out for the system to detect the entire communication of the individual, who tries to show a letter from the sign language alphabet by coming across the camera by hand in real-time with high accuracy ratio. The "Turkish Sign Language Dataset" on the Kaggle website has been used in the project for training our system. The dataset contains images of 26 letters that were taken from different angles and made by different individuals. While there were approximately 4000 images, approximately 1600 of them were used for training. Since some images were not of sufficient quality for training, approximately 100 photographs taken by ourselves have been added to the dataset. Some of the sample images in our dataset are given in Figure 2.

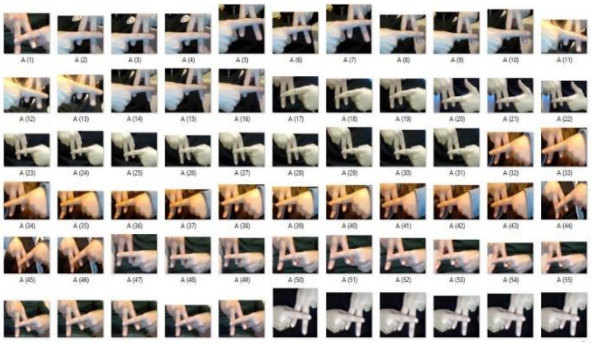

Figure 2. Some images from dataset

Then, the editing or extraction of data that was not sufficient for training and deemed unnecessary in the selected dataset was done. Upon completion of the necessary procedures, approximately 4000 images were reduced to approximately 1600 images. In addition, in the parts that do not seem sufficient, the images we took (approximately 100) were added to the dataset. Thus, the dataset has been prepared for the training. After this stage, the photos were labeled before feeding to the deep learning model for training.

3.2. Data Labelling

In order to detect objects in real time in front of the camera, the data must first be introduced to the computer. The photos in the edited dataset were tagged and introduced to the computer to which class it belongs to. This process was done using OpenCV library. After this process, the XML file of each photo was created. However, the computer cannot read XML file during training. Therefore, each XML file has been converted into a single CSV file. This was done using a Python script. Each line in the CSV file contains XML information for a different photo. As a result, the dataset becomes ready for training.

3.3. Training

For this step, the libraries required for the training of the system were installed by creating a virtual environment in Anaconda IDE. Among the libraries to be used, the most important for the training of the system is the TensorFlow library. Tensorflow is an open source deep learning library. Being open source allows the problems encountered to be solved easily via customizing the code. One of the biggest advantages of the library; with the APIs it offers, the platform allows us to use one or more CPUs and GPUs independently. In addition, the library's development with Python maximizes its compatibility for our system. Tensorflow contains many configuration files and allows these files to be manipulated for training the system. In this way, the configuration files were prepared in accordance with our system and made ready for training. Tensorflow offers weight files based on the number of data in the system and the training conditions (speed, accuracy rate, etc.). After the weight files were integrated into the system, the number of letters and symbols to be used in the project were also introduced to the system. After this process, the configuration file was edited and the number of output classes was written. This number is 26 in our project.

After then, the training of the system began. The training period lasted approximately 10 hours and the prediction time was less than 2 seconds. Tensorflow displays the loss function after each transaction, informs about the stage of the training and makes it easier for us to make a decision to stop it.

In addition, Tensorflow shows all stages of training on the graph in a web interface running on localhost. Training continues until you achieve the lowest loss rate. When the lowest loss rate is obtained, the training is stopped and the “frozen inference” graphic formed as a result of the training is exported for the operation of the system. The default name of the file is “frozen_inference_graph.pb”. By means of this file, the performance of the training and the accuracy rates were observed.

3.4. Testing

After the training, it is very important to test and check the accuracy of the system. If error ratio is high in real time detection, the system may need to be re-trained and the model may be re-generated. In this step, image quality and resolution are one of the factors that increase the accuracy rate. Two different cameras were used during the test. In addition, the performance of the system was measured by testing hand movements of different people. The effect of complex or empty background during hand movement have been observed on the performance of the system. All 26 letters are statically and dynamically tested. The system has been tested using photos prepared for testing beforehand. Then, real-time tests were carried out using the camera. Figure 3 shows some of the real-time test results.

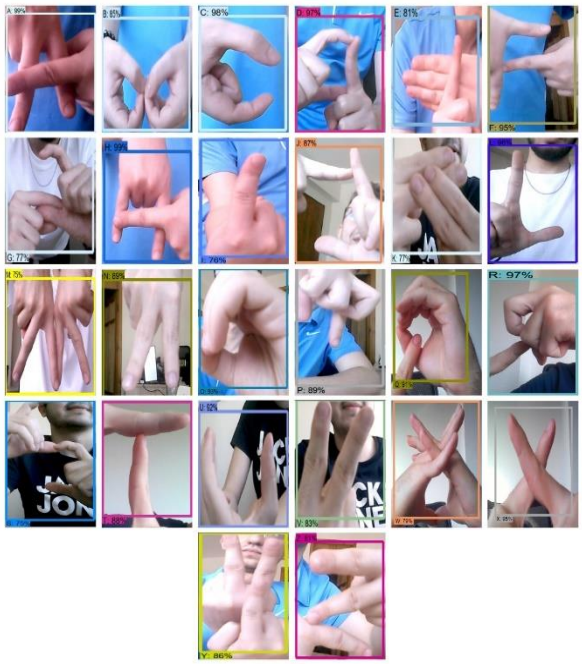

Figure 3. Some of the real-time test results

The statistical analysis was made according to the estimation results produced by the system. There are two cases, the system either shows the correct letter or the wrong letter. While TP (True Positive) shows the result that the model predicts the positive class correctly, FN (False Negative) shows the result where the model incorrectly predicts the negative class. Precision shows how many of the values we predict positively are actually positive. Recall shows how many of the trades we should have predicted positively. Precision and recall values do not give a meaningful comparison when calculated separately. Therefore, it would be more correct to evaluate these two values together. For this, we need to calculate the F-Score value. F-Score is the harmonic mean of precision and recall values.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Table 1 shows the accuracy, the F-Score, true positive and false negative values for each letter that was introduced to the trained system during testing. The results were obtained by showing 50 signs from each letter in front of the camera.

Table 1. Test Results Analysis Table

Sign	TP	FN	Accuracy	F-Score
A	98%	2%	98%	98%
B	80%	20%	80%	88%
C	94%	6%	94%	96%
D	80%	20%	80%	88%
E	88%	12%	88%	93%
F	92%	8%	92%	95%
G	74%	26%	74%	85%
H	86%	14%	86%	92%
I	90%	10%	90%	94%
J	80%	20%	80%	88%
K	80%	20%	80%	88%
L	94%	6%	94%	96%
M	76%	24%	76%	86%
N	76%	24%	76%	86%
O	80%	20%	80%	88%
P	88%	12%	88%	93%
Q	74%	26%	74%	85%
R	92%	18%	92%	95%
S	94%	16%	94%	96%
T	90%	10%	90%	94%
U	88%	12%	88%	93%
V	88%	12%	88%	93%
W	86%	14%	86%	92%
X	96%	4%	96%	97%
Y	88%	12%	88%	93%
Z	92%	8%	92%	95%

The system has been tested in real-time with a total of 1300 images. According to the results, the overall accuracy rate of the system was calculated as 88%. The TP rate was 87% and the FN rate was 13%. During the test, it was observed that the delay time of the camera was 28 milliseconds. This indicates that there will be no freezing or noticeable delay when using the system. Thus, users will be able to communicate between them fluently and smoothly.

4. Performance Measurement

To analyze the performance of the system, the infographics presented by TensorFlow in the local environment were examined after the training phase. Looking at the graphs given in Figure 4, as the iterations increase, accuracy approaches 1 while the loss function approaches 0. This shows that the more iterations there are, the higher the accuracy rate of training. While the calculated accuracy rate is 88%, the total loss is 0.1.

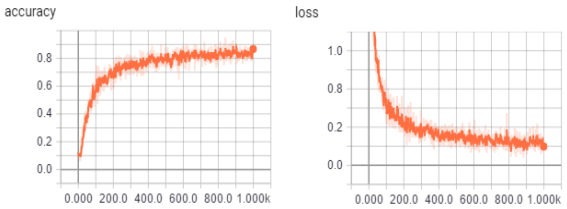

Figure 4. Accuracy and Total Loss Graph

The distributon charts of bias and weight are given in Figure 5. The X axis indicates time, the Y axis indicates values. These charts are a top view of histogram charts. It would be more accurate to examine the histogram graphs for interpretation.

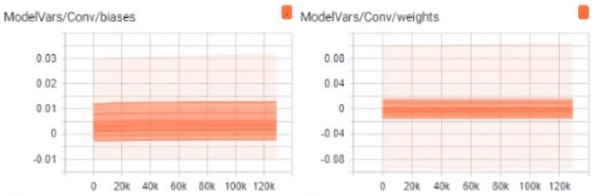

Figure 5. Biases and Weights Distribution View

Histogram graphs are shown in Figure 6. There are three axes in the histogram. According to the figure, X-axis values (depth), Y-axis time, Z-axis shows the intensity (frequency) of values represented on the Y-axis. Histograms become darker as the depth in the X-axis increases. Taking a higher value on the Z axis means the vector produces a value closer to the specified value.

Figure 6. Biases and Weights Histogram

The results in Table 2 were obtained when the methods in the other 2 articles for the recognition of Turkish sign language by image processing were compared.

Table 2. Comparison Table with Other Studies

	Dataset	Image Processing	Models	Accuracy
Çelik and Odabaş (2020)	Images from 200 videos	Non-real time	CNN without preprocessing	53%
			CNN with preprocessing	91%
			CNN+LSTM with preprocesing	97%
Unutmaz et al. (2019)	4320 images from videos	Real time	CNN	81%
			SVM	71%
			KNN	61%
			DT	54%
		Non-real time	CNN	93%
			SVM	82%
			KNN	80%
			DT	75%
Our proposed method	1700 images	Real time	Faster-R-CNN	88%

In Table 2, the accuracy values for the methods used in Unutmaz et al. (2019) were given for the session in which the training was performed with the %5 portion of the whole data set. Because, real time performances of those methods were also evaluated using the the same portion value for the training data set. Çelik and Odabaş (2020) used the manually separated hand gesture frames from videos for recognition using CNN model which was a non-real time process. The accuracy of the CNN model increased when they applied preprocessing to extract the hand objects from the frames. However, our method achieved %88 accuracy rate in real time without any preprocessing.

5. Conclusions

As a conclusion, we proposed a system that converts 23 letters in the Turkish Sign Language Alphabet and 3 letters (Q, W, X) in English Sign Language into text using image processing and deep learning methods. Our system which performs this process in real time used the Faster R-CNN architecture unlike other systems in the literature. In addition, the development of the system with Python increases its comprehensibility, while the use of open source libraries facilitates its accessibility. While the system takes the hand image that mimics the sign language alphabet as input, it shows the letter that looks like the hand image as output with its accuracy rate. When we look at performance measures such as Accuracy rate, F-Score and Loss value, our system seems to be successful. The developed system will facilitate communication between people who do not speak sign language and individuals with hearing disabilities. Thus, people with hearing impairments will be prevented from isolating themselves from society and their self-confidence will be increased.

Acknowledgment

The dataset used for training in this study is available at https://www.kaggle.com/feronial/turkish-sign-languagefinger-spelling.

References

Ashish S. N., Aarti G. A., 2016. Sign language recognition using image based hand gesture recognition techniques. 2016 Online International Conference on Green Engineering and Technologies (IC-GET), 19 November 2016, Coimbatore, India, pp.1-5.

Çelik Ö., Odabaş A., 2020. Sign2Text: Konvolüsyonel Sinir Ağları Kullanarak Türk İşaret Dili Tanıma. European Journal of Science and Technology, 19, 923-934.

Gaikwad, S., Shetty, A., Satam, A., Rathod, M., Shah, P., 2019. Recognition of American Sign Language using image processing and machine learning. International Journal of Computer Science and Mobile Computing, 8(3), 352-357.

Kumarage, D., Fernando, S., Fernando, P., Madushanka, D., Samarasinghe, R., 2011. Real-time sign language gesture recognition using still-image comparison & motion recognition. 6th International Conference on Industrial and Information Systems, 10 October 2011, Kandy/Sri Lanka, pp.169-174.

Pramada S., Saylee D., Pranita N., Samiksha N., Vaidya S., 2013. Intelligent sign language recognition using image processing. IOSR Journal of Engineering (IOSRJEN), 3(2), 45-51.

Sandhya, A., Ananya R., 2018. Recognition of sign language using image processing. International Journal of Business Intelligence and Data Mining, 13(1-3),163-168.

Shaoqing R., Kaiming H., Ross G., Jian S., 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137-1149.

Shivashankara S., Srinath S., 2018. An American sign language recognition system using bounding box and palm features extraction techniques. International Journal of Recent Technology and Engineering (IJRTE), 7(4s), 492-505.

Tamura, S., Kawasaki, S., 1988. Recognition of sign language motion images. Pattern Recognition, 21(4), 343-357.

Tan Y.S., Lim K.M., Tee C., Lee C., 2020. Convolutional neural network with spatial pyramid pooling for hand gesture recognition. Neural Computing and Applications, Published online: 15 September 2020.

Unutmaz B., Karaca A.C., Güllü M.K., 2019. Turkish Sign Language Recognition Using Kinect Skeleton and Convolutional Neural Network. 27th IEEE Signal Processing and Communication Applications Conference, April 2019, Sivas, Turkey, pp.1-4.

Vargas L.P., Barba L., Torres C.O., Mattos L., 2011. Sign Language Recognition System using Neural Network for Digital Hardware Implementation. Journal of Physics: Conference Series, 274 (2011) 012051, 1-7.