Real Time Sign Language Recognition Using Neural Networks

Ali SU
Electronics and
Communication Engineering
lstanbul Technical University
Istanbul, Turkey
su16@itu.edu.tr

Yasser SULAIMAN
Electronics and
Communication Engineering
lstanbul Technical University
Istanbul, Turkey
sulaiman17@itu.edu.tr

***Abstract*— Sign Language is the language of communication that is used by the deaf people like every other language. With the development of image recognition technology, many studies have appeared on the issue of determining sign language using cameras. In this project, we aimed to use artificial neural networks for real time sign language recognition in digital environment. After discussing the architecture and theory, the project was tried to be implemented in digital environment using the Python programming language. The data set of Gesture Image Normal and Preprocessed images was added to the code, and after a particular number of epochs, over 99% of their precision was achieved.**

Keywords—Deep Learning, Neural Network, Sign Language

I. INTRODUCTION

The main language of the Deaf community, their preferred medium of contact, is sign language. No universal sign language except International Sign (IS) or Gestuno, which are used at international conferences organized by deaf people, is selected for the purpose of generalizing language [1].

Many methods such as reading from lips and spelling with fingers have been used in sign languages from past to present. Unfortunately, today the deaf community has difficulty communicating with other people who do not know sign language. For this reason, sign language recognition (SLR) is a multidisciplinary research area that includes computer vision, pattern recognition, linguistics, and natural language processing [2].

We can divide sign language recognition studies into glove-based systems using data gloves and vision-based systems. Glove systems have sensors attached to the glove, such as the combined accelerometer and surface electromyographic sensor, to capture the hand and finger movement and rotation. The several efforts have been made to interpret the hand gestures, especially the signals changing over time, which are used as an essential tool for most works by the Hidden Markov Model (HMM). The Chinese sign-language signals are understood by a system with two gloves and three-position trackers as input devices, and a fuzzy decision tree as a classification model. It reaches 91.6% recognition accuracy with a 5113 sign vocabulary. [3]. Although glove-based systems are less affected by external parameters, they are not currently suitable for general use in terms of cost.

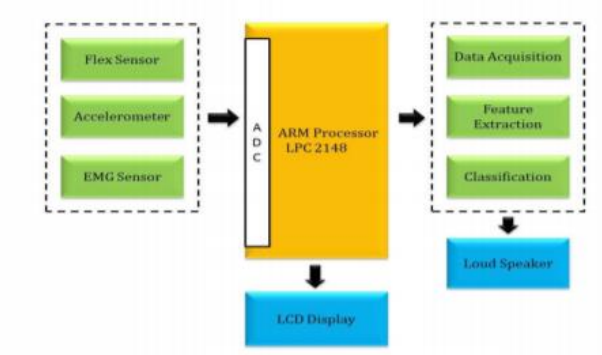

***Figure 1**: A Hand-Talk glove block-diagram consists of one flex sensors for each finger, one 3-axis accelerometer, text to speech conversion module, LCD display and three sEMG sensors with one being a reference electrode [4].*

On the other side, the vision-based system offers a more natural and contact-free solution, for example, the camera to detect information on human actions and their environment. Intelligent processing makes it difficult to model due to complex backgrounds and occlusions. Some previous studies have used a system with reconfigurable FPGA hardware [5], colored gloves [6] or many cameras for precise hand detection, segmentation, and identification to improve the robust efficiency of the vision-based approaches.

II. Dataset

The Sign Language Gesture Images Dataset has been used for training our model. The original dataset contained 37 different categories each with 1,500 images. The dataset consists of numbers 0 through 9 and all 26 alphabets in addition to the space sign. in order to lower the computation cost of the training process, we did not train the model for the numbers but only for the characters. for J and Z characters we eliminate them from the dataset since recognition of these characters requires motion detection. moreover, we added the 'Nothing' category to the dataset since there was no such a class in the original one. finally, we end up with 37500 images belong to 25 different classes.

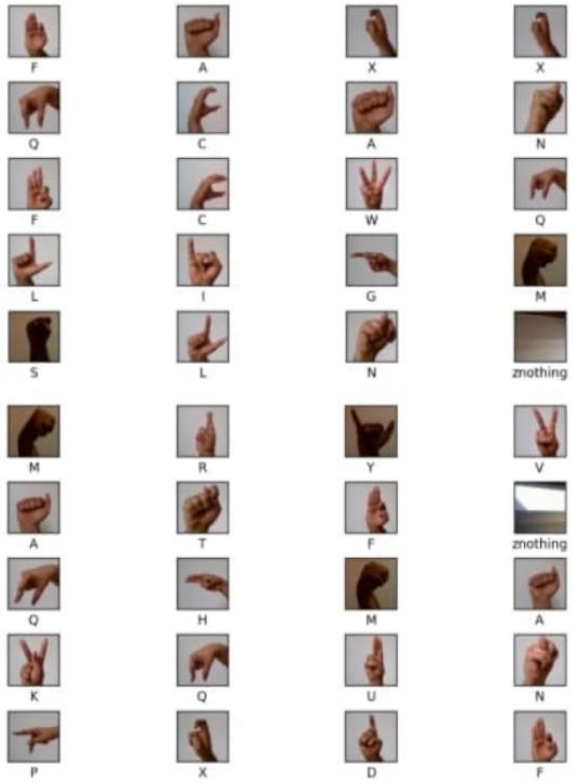

Figure 2: *Samples From Train Dataset*

III. Methodology

Considering the nature of our dataset we have decided to utilize convolutional and pooling layers in our model which are the most suitable layers for extracting features from images. using convolutional layers makes a significant reduction in the number of the parameters to be learned which allows building deeper networks with fewer parameters. in addition, convolutional neural networks (CNNs) are better in terms of feature extraction compared to fully connected networks. Figure 3 shows how convolutional layers work. We prefer to use the following formula to determine the spatial dimensions of the output of the convolution layers:

$$\frac{(V-R)+2Z}{S+1} \qquad \textit{(Eq. 1)}$$

Where V is the size of the input volume, S is the value of the phase, Z is the sum of zero padding collection, and R is the size of the receptive field.

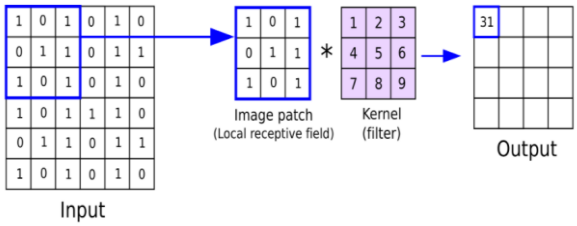

Figure 3: *Convolutional Layer [7]*

The pooling layers on the other hand down sample the data without losing any important features. in other words, using pooling layers makes it possible to obtain the same amount of features with lower dimensions. Maxplooing operation is illustrated in Figure 4.

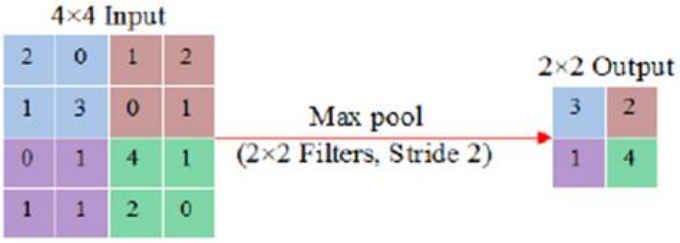

Figure 4: *Max Pooling [8]*

A. Proposed Model

The structure of our model is summarized in Figure 5. the first layer of the model is a convolutional layer with the input of shape (50,50,3) and the output layer is a fully-connected layer with an output shape of (None, 25) which represents the different 25 classes of our dataset. the Rectified Linear Unit (ReLU) activation function is used in input and all intermediate layers. The primary reason for using the ReLU function instead of other activation functions is that not all the neurons are activated simultaneously, meaning that some of the weights and biases are not updated during the backpropagation process which makes the training even faster [9]. ReLU function is illustrated in Equation 2:

$$g(y)=\begin{cases}0 \text{ for } y<0\\ y \text{ for } y\geq 0\end{cases} \qquad \textit{(Eq. 2)}$$

A softmax function that converts the scores obtained through the forward propagation process into normalized probability distribution is used in the output layer in order to define the expected class according to the highest probability. Softmax function could be demonstrated as in Equation 3:

$$\alpha(c)_j = \frac{e^{c_j}}{\sum_{k=1}^{K} e^{c_k}} \quad \textit{(Eq. 3)}$$

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 48, 48, 16)        448
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 46, 46, 16)        2320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 44, 44, 16)        2320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 22, 22, 16)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 20, 20, 32)        4640
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 18, 18, 32)        9248
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 6, 6, 64)          18496
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 2, 2, 64)          36928
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 128)               32896
_________________________________________________________________
dense_1 (Dense)              (None, 25)                3225
=================================================================
Total params: 156,697
Trainable params: 156,697
Non-trainable params: 0
```

Figure 5: *The structure of our model.*

B. *Training*

We have split our dataset into three different portions as follows: 80% of the data used for training, 10% of the data used for testing, and 10% of the data used as validation dataset. Default batch size of 32, Adam optimizer with default learning rate of 0.001 and categorical cross-entropy loss function has been used for compiling the model. the data has been normalized and have been trained for 20 epochs over a local NVIDIA GeForce GTX 1660 TI GPU. From Figure 6 we can see that the training and validation accuracy increases by the progress of the training process while Figure 7 shows the decrease of the related losses.

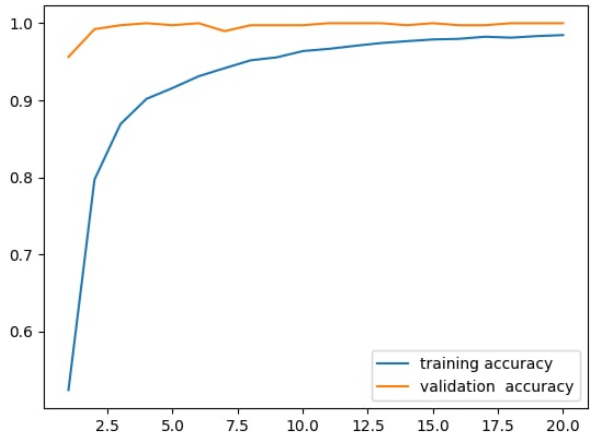

Figure 6: *Validation & Training Accuracy*

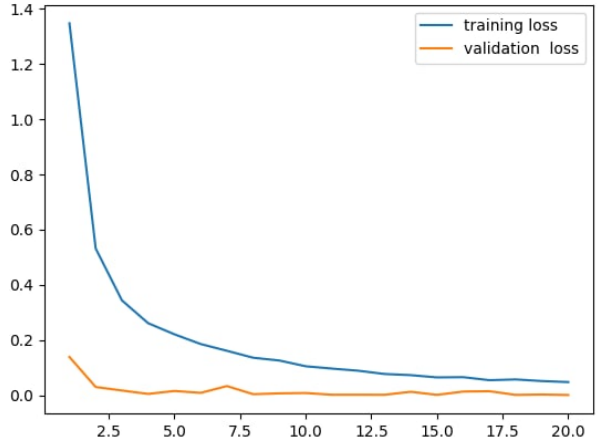

Figure 7: *Validation & Training Loss*

IV. Result

The first model we designed was good at recognizing non-similar characters but it had problems with classifying similar signs such as A and T, K and V correctly. we noticed that shallow networks cannot differentiate between images with similar features so we decided to use a deeper network in order to extract more complex features that are unique for each image. our final model which is consists of 14 layers and 156,697 parameters could successfully classify all of the characters with a training and testing accuracy of 99 percent. some output examples are shown in Figure 8.

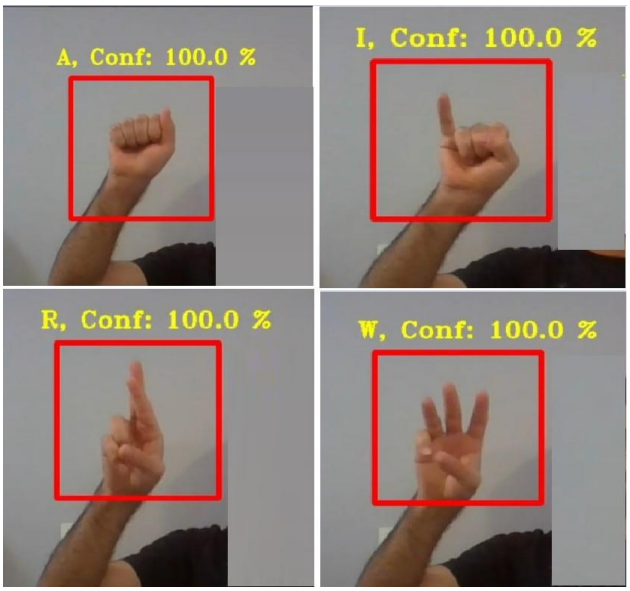

Figure 8: *Real-Time Results*

V. Conclusion & Future Work

In this project, we implemented an artificial neural network for real-time sign language recognition applications. We used Convolutional Neural Network (CNN) as a learning method. With these parameters, we obtained an accuracy of 99% at the end of 20 epochs. As a result, we were able to achieve the targeted accuracy rate. for future work, we will try to use object detection algorithm such as YOLO to detect the position of the hand instead of the predetermined region used in the current version of the model. in addition, we will work on extracting the hand from the background before feeding it to the network for more accurate results.

References

[1] Rubino, F. H. (1975). Gestuno.International sign language of the deaf. Cblisle: British Deaf Association.

[2] Aran, O. (2008). Vision based sign language recognition: modeling and recognizing isolated signs with manual and non-manual components'. Istanbul: Bogazici university.

[3] G. Fang, W. Gao, and D. Zhao, 2004, "Large vocabulary sign language recognition based on fuzzy decision trees," IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, vol. 34, no. 3, pp. 305–314.

[4] Anetha, K., & Parvin, J. (2014). Hand Talk-A Sign Language RecognitionBased On Accelerometer and SEMG Data. International Journal of Innovative Research in Computer and Communication Engineering, 2.

[5] Vargas, Lorena & Barba J., Leiner & Torres Moreno, Cesar & Mattos, Lorenzo. (2011). Sign Language Recognition System using Neural Network for Digital Hardware Implementation. Journal of Physics: Conference Series. 274. 012051. 10.1088/1742-6596/274/1/012051.

[6] T. Starner, J. Weaver, and A. Pentland, 1998, "Real-time American Sign Language recognition using desk and wearable computer based video," IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 12, pp. 1371–1375

[7] Anh H. Reynolds, "Convolutional Neural Networks (CNNs)," Anh H. Reynolds, 15-Oct-2017. [Online]. Available: https://anhreynolds.com/blogs/cnn.html .

[8] Sakib, Shadman & Ahmed, & Jawad, Ahmed & Kabir, Jawad & Ahmed, Hridon. (2018). An Overview of Convolutional Neural Network: Its Architecture and Applications. 10.20944/preprints201811.0546.v1.

[9] D. G. Dishashree, "Activation Functions: Fundamentals Of Deep Learning," Analytics Vidhya, 19-Jul-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/#:~:text=The%20main%20advantage%20of%20using,neurons%20at%20the%20same%20time.&text=Due%20to%20this%20reason%2C%20during,neurons%20which%20never%20get%20activated .