Recognition of Colombian Alphabeth in Sign Language Using Deep Learning Techniques

E Arrieta-Rodríguez, R E Monterroza-Barrios, P L Torres-Alvarez, G E Castro-Lozano
Universidad del Sinú Elías Bechara Zainum Cartagena
E-mail: investigacionsistemas@unisinucartagena.edu.co

Abstract. This work presents the development of an algorithm for sign language alphabet recognition Colombian in images, using the deep learning technique. For this, a data set consisting of each letter of the alphabet from A up to Z and a total of 4043 images with a resolution of 150x150 pixels of 3 channels, a red neural was built convolutional with 256 layers, with an accuracy of 75%.

1. Introduction
Sign language is a language used throughout the world by audio-impaired people, but with notable differences between countries, specifically in this work it is covered in Colombia [8]. People with hearing disabilities present some problems at a personal, social, and work level, which make it difficult for them to fully include society, for this reason this work proposes the construction of a tool that facilitates interaction between audio-impaired people and the rest of the society.

2. Related word
Numerous investigations have been carried out on the incorporation of sign language in computer resources in the educational field, both in our country and in the world, this section mainly focuses on national and Latin American studies and experiences, due to the importance of in this dissertation he awards the social context that characterizes the educational field where Information Technology -IT- is incorporated.

One project titled "Recognition of sign language using image processing" developed by Ananya Roy and Sandhya Arora, January 2018. It proposes the system recognizes the alphabet of sign language by calculating the distance metric of Bhattacharyya between the histograms of the captured image, processed and the stored image and outputting that alphabet whose image histogram has a smaller Bhattacharyya distance with the histogram of the captured image [1].

Also the project "Automatic voice recognition method for the classification of vowels to Colombian sign language." developed by Andres Arias and David Rubiano, 2018. It proposes the system recognizes the vowels of the Colombian alphabet by means of audio in order to recognize the letter by audio and return the vowel by Colombian sign language [1].

Additionally, we found that in Mexico City, D.F., at the Instituto Técnico Nacional in the computing research center, the editor Fausto Pável Priego Pérez, carried out a project with the
name of *Recognition of images of Mexican sign language*. This work proposes a system of two main modules which consist of learning and recognition. Use a video camera device (Kinect device) to detect the pattern or sign of each word; Two methods are used for the recognition of the sign, in the first one requires some type of specific gloves to recognize the sign. In the second, the Kinect device is used for recognition [6].

In 2012 at the Córdoba Faculty of the Technological University - Argentina, he presented a research project based on the Kinect system of the Xbox game console, an interpreter and translator software was produced for both sign language and natural language, so allows users to interact in different ways, it has a sign for each word and also for each letter in case you want to spell a non-existent word [4].

Financially, several works are identified, such as [2],[7], [5] which are related to sign language, with the application of different techniques from computer science.

3. Methods and Materials
This section describes the process of collecting, organizing and finally building the database of the sign language alphabet, through photographic registration. It also describes the process of modeling, design, normalization, division of the data and the diagram of the data.

3.1. Data collection
Taking into account that a database of images with the sign alphabet was not available in a local context, it was decided to build it manually. For this, the taking of photos was done, using cell phone cameras, this was necessary to meet the minimum amount required for the training of the model. The two smartphones that were used were the ZenFone Max (M2) and the Neffos y5s. The collection was made by the two researchers, and the participants who served as models to take photos of their hands indicating each letter of sign language. 4043 photos were taken for training and testing of the model.

3.2. Database Construction
Following this, the segmentation process by password was carried out; This process is also called the labeling process, where the names of each folder represent a letter of the Colombian alphabet in sign language. 4043 photos were taken of the 26 letters of the alphabet in sign language, in the data set there are two types of images due to the difference in dimensions that was caused by the use of the two devices for taking the images. To solve this problem, the data dimensions are standardized in the model.

3.3. Modeling
Figure 1 presents the design of the algorithm for the detection of sign language, this raises the segmented development in two phases, the first is the creation of the data set and the second correspondence to the construction stage of the neural network model convolutional [3].

3.4. Data Normalization
It is useful to normalize the data to make learning easier for machine learning. By normalizing we mean putting all the data on the same scale. It was necessary to take the data matrix and perform a normalization by applying a division of the matrix containing the training data between 255 (training data / 255 the maximum value of a pixel). Keras was used to process the images at the scale of data pixels 1/255, this normalization allowed the matrix operations that were carried out in training to be much faster and more efficient both for the multiplication that is carried out with the weights that are initialized and for the update of weights with respect to the level of error to be decreased.
3.5. Split Data

The Split technique is applied, which operates with a division of the total data in a distribution of 80% for training data and 20% for evaluation, the objective of having these divided data was relevant to us to corroborate with the evaluation data, if our generalized models with unknown data (show Figure 2).

3.6. Load Data

With the data set already built and saved in the Google drive tool, using the unique Google Colaboratory libraries to load the data into it. In it, the Python code was executed with the TensorFlow and Keras libraries for the construction and evaluation of the CNN (Convolutional Neural Network) model.

4. Architecture

In the architecture of the model the functions that helped to build the convolutional neural network are defined, then it will be briefly explained how these functions (Show Figure 3).

(i) Sequential(): This function is a stack of linear layers, which are created by passing an instance list of layers to the constructor.

(ii) add(2D convolution): This layer creates a convolution core that is interlaced with the input layers to produce an output tensor. When using this layer as the first layer in a model,
certain keywords are provided which are filters, kernel size, strides, padding, input shape and activation.

- Filters: is the dimension of the convolution output.
- Kernel size: specifies the dimensions of the input image.
- Strides: convolution strides far and wide.
- Padding: same results in padding the input so that the output is the same length as the original input.
- Input shape: indicates that the expected input will be batches of vectors of number of dimensions.
- Activation: can be used through one activation layer or it can be supported by all layers.

(iii) MaxPooling2D: Maximum grouping operation for spatial data. MaxPooling2D is a 3D shape tensor (weight, height, channel) the height and weight dimensions tend to decrease as we enter the hidden layers of the neural network. Where Units means Pool size: tuple of 2 integers that reduces the scale (vertical, horizontal). (2, 2) will halve the input in both spatial dimensions. If only an integer is specified, the same window length will be used for both dimensions.

(iv) Flatten: Converts the elements of the input image array to a flat array.

(v) Dense: With this instruction we add a hidden layer of the neural network. As Units means the dimensionality of the output space.

(vi) Dropout: Dropout is randomly setting a rate fraction of the input units to 0 or 1 on each update during the training time, which helps avoid overfitting.

(vii) Compile: Set up the model for training.
- Optimizer: instance of the optimizer.
- Loss: instance loss function, if the model has multiple outputs, you can use a different loss in each output passing a dictionary or a list of losses.
- Metrics: List of metrics that the model will evaluate during training and testing.
Figure 3. Model Diagram
5. Training and Test Model
Once the suitable data set has been obtained and also the machine learning technique that will be used to develop the letter classifier for sign language. The convolutional neural network technique was used, the model training was carried out with 80% of the data set and 20% of the set was used for the validation of the model. In the training it was done in 15 epochs (epoch) that are the times that the model will be trained, with 100 steps per epoch (sample batches) it is the one that says when one epoch ends and the other begins.

![Model accuracy](image)

**Figure 4.** Accuracy

Previous in Figure 4 we can see the progress of the model’s accuracy as the epochs are iterated, discriminated between the training (train) and the validation (test) of the model.

Next in Figure 5 shows the loss of the model in the passage of each epoch, both in the training process and in tests. As a result, a close loss in training and testing is obtained, and finally at time 12 the model reaches a very similar loss.

![Model loss](image)

**Figure 5.** Loss

6. Results
After training the machine learning model, it is necessary to evaluate the performance of the classifier to determine its viability; one of the most used metrics when evaluating an algorithm is
the confusion matrix. This gives us relevant information about the performance of the classifier, since from it other metrics are extracted that give true information about it.

The confusion matrix allows the visualization of the performance of an algorithm and helps to validate the successes in the prediction and the frequency of errors. The variables H, N and Z (letters of the alphabet) have low correct answers, since in sign language these images have similarities with others (show Figure 6).

Figure 6. Confusion matrix

Figure 7 shows precision, recall, f1-score, support in detail. The precision provides information about their performance of how many they have captured, recall gives information from a classifier of how many have failed, f1-score is the weighted average of precision and recovery, and support were the number of images that were used for each letter to make matrix.

The accuracy of the model is 0.75 percent precision, failing more where the f1-score has the lowest value. This implies that the model has to be trained with a greater number of images to obtain a higher accuracy result.

7. Conclusions
At the end of the algorithm tests, we conclude that the code works with a margin of error of approximately 25

- Initially, a manual data set was built which was used to train the model.
- The neural network was then trained with the training data set, which then led to the evaluation of the model to wait for the results given in a classification report and a confusion matrix.
- Finally, the validation stage is reached. It was a work in conjunction with the test stage, since after observing the test results it was possible to analyze results to determine the best behavior of the training parameters.
This work is presented as an initial approach to the use of Deep Learning techniques applied in a real problem, such as communication in sign language, in search of achieving an inclusive society for this population.

References
[1] Arora, S., Roy, A.: Recognition of sign language using image processing. International Journal of Business Intelligence and Data Mining 13(1-3), 163–176 (2018)
[2] Castro Estévez Luis Alberto, C.G.K.K.: Herramienta de apoyo para la interpretación de lenguaje de señas mexicano (hilsem) (2015)
[3] Crisvill: Avances en redes neuronales. medium (2016), https://medium.com/espanol/avances-en-redes-neuronales-705c2efe53d2
[4] Matías Reggi, Fabián Bertetto, E.R.: Inventaron software que interpreta lenguaje de señas. buendiario (2015), https://www.buendiario.com/inventaron-software-interpreta-lenguaje-senas
[5] de Oliveira Andrade, R.: Software convierte voz en lenguaje de señas para sordos. scidev.net (2014), https://www.scidev.net/america-latina/comunicacion/noticias/software-convierte-voz-en-lenguaje-de-se-as-para-sordos
[6] Pérez, F.P.: Reconocimiento de imágenes del lenguaje de señas mexicano. México, DF: Instituto Politécnico Nacional (2012)
[7] Saavedra, Y.: Microsoft planea usar kinect para interpretar el lenguaje de señas. hipertextual (2013), https://hipertextual.com/2013/07/kinect-lenguaje-senas
[8] Salazar, M.C., Molina, D.B.: Instituto nacional para sordos–insor-observatorio social

| Classification Report | precision | recall | f1-score | support |
|-----------------------|-----------|--------|----------|---------|
| A                     | 0.96      | 0.67   | 0.79     | 36      |
| B                     | 0.94      | 0.86   | 0.90     | 35      |
| C                     | 0.92      | 0.41   | 0.56     | 27      |
| D                     | 0.77      | 0.65   | 0.70     | 31      |
| E                     | 0.92      | 0.71   | 0.80     | 31      |
| F                     | 0.66      | 0.94   | 0.77     | 31      |
| G                     | 0.98      | 0.60   | 0.72     | 30      |
| H                     | 0.50      | 0.22   | 0.31     | 35      |
| I                     | 0.66      | 0.74   | 0.70     | 31      |
| J                     | 0.68      | 0.97   | 0.80     | 39      |
| K                     | 0.65      | 0.69   | 0.67     | 32      |
| L                     | 0.76      | 1.00   | 0.86     | 28      |
| M                     | 0.50      | 0.86   | 0.63     | 28      |
| N                     | 0.53      | 0.24   | 0.33     | 34      |
| O                     | 0.90      | 0.93   | 0.92     | 30      |
| P                     | 0.84      | 0.55   | 0.67     | 29      |
| Q                     | 0.98      | 0.90   | 0.90     | 30      |
| R                     | 0.79      | 0.79   | 0.79     | 33      |
| S                     | 0.62      | 0.94   | 0.74     | 31      |
| T                     | 0.73      | 0.93   | 0.82     | 29      |
| U                     | 0.80      | 0.93   | 0.86     | 30      |
| V                     | 0.81      | 0.97   | 0.88     | 31      |
| W                     | 0.88      | 0.93   | 0.90     | 30      |
| X                     | 0.78      | 0.78   | 0.78     | 27      |
| Y                     | 0.65      | 0.94   | 0.77     | 36      |
| Z                     | 1.00      | 0.46   | 0.63     | 26      |

| accuracy              | 0.75     | 801    |
| macro avg             | 0.77     | 0.75   | 0.74   | 801    |
| weighted avg          | 0.77     | 0.75   | 0.74   | 901    |

Figure 7. Metrics