American Sign-Language Classifier

Gonzalo Vega Pérez
Robotic Software Engineering
URJC
Fuenlabrada, Spain
g.vega.2020@alumnos.urjc.es

Jaime Avilleira García
Robotic Software Engineering
URJC
Fuenlabrada, Spain
j.avilleira.2019@alumnos.urjc.es

Julia López Augusto
Robotic Software Engineering
URJC
Fuenlabrada, Spain
j.lopeza.2020@alumnos.urjc.es

Alan Josué Melgar Fuentes
Robotic Software Engineering
URJC
Fuenlabrada, Spain
aj.melgar.2023@alumnos.urjc.es

***Abstract*—The objective of this document is to solve a problem using a Machine Learning technique learned along this subject.**

In this case, the approach chosen is trying to help people having hearing loss. To deal with that, it has been decided to create a neural network to classify and recognize gestures using Deep learning.

Keywords— neural network, sign-language, classifier, recognize, images, dataset.

I. Introduction

Nowadays, machine learning is booming, and there are many applications related to this topic that allow automation, assistance to society, and the enrichment of knowledge. To continue contributing to society, we have been exploring potential proposals. Ultimately, we chose to assist people with hearing loss by learning and interpreting sign language, enabling them to communicate with a larger audience.

Within this context, the solution we found was to create a classifier to recognize the gesture and translate it into the corresponding letter. For those who may not be aware, sign language is not a universal language. This complexity posed a challenge for our solution

To address this issue in any language, we have chosen to employ supervised deep learning techniques.

II. Problem statement

To solve the problem, we decided to divide it into subproblems:

A. WHAT IS SIGN LANGUAGE?

To delve deeper into this topic, it is important to understand the context and its history:

Sign language is a visual and gestural communication system used by deaf or hard-of-hearing individuals to express and receive information. Instead of relying on speech and sound, it is based on gestures, hand movements, facial expressions, and other bodily signs. Each country or region may have its sign language, and these languages are entirely natural with their own grammatical rules and structures.

Sign languages have naturally existed in deaf communities for centuries. There is no specific origin pinpointed as they have evolved organically in diverse communities worldwide. Recognition and appreciation of sign languages as legitimate languages have grown over time.

There is no fixed number of sign languages globally, as each country or region may have its unique system. It is estimated that there are several hundred sign languages worldwide.

The number of sign language speakers varies significantly. There are no precise statistics for all sign languages, but it is estimated that millions of people worldwide use sign languages as their primary means of communication. The exact count depends on the deaf population in each country or region and the prevalence of sign language usage in those communities.

B. CHOSEN LANGUAGE

The absence of a common sign language has led us to investigate which language to use to address the problem.

As mentioned in the statement, each language has its sign language, and due to the inability to create our dataset, we had to explore and select a language based on existing resources.

After a tedious search for viable datasets to fulfill the desired task, we decided to use American Sign Language (ASL) as it is the most extensive language with large datasets from which we could obtain optimal results. It would have been interesting to do this in our native language, Spanish, but the datasets were less numerous.

A very clear example to highlight the difference between sign languages in different languages is as follows:

To say "hello" in Spanish Sign Language, it is typically done by raising the hand and moving the fingers as if waving.

To say "hello" in American Sign Language (ASL), one raises the open hand to shoulder height and makes a forward and backward motion toward the interlocutor [1].

III. Information Retrieval

After selecting the desired language, it is important to search for the necessary information to address the problem at hand:

A. Dataset Search

There are various types of datasets, specifically, we will need to work with images, each designed to address specific needs in the field of Machine Learning. Some common types of image datasets include:

- **MNIST:**

  A dataset containing grayscale images of handwritten digits (0 to 9), commonly used for digit classification problems.

- **CIFAR-10/100**:

  Datasets containing small color images of 32x32 pixels, categorized into 10 and 100 classes respectively, used for object classification problems.

- **ImageNet**:

  One of the largest datasets, including millions of high-resolution images categorized into thousands of classes, used for classification and object detection tasks.

- **COCO:**

  A dataset containing images depicting everyday scenes with multiple objects in various contexts, used for object detection and segmentation tasks.

B. Types of model:

To address image classification, various models, and deep learning architectures can be employed. Some common options include:

- **Convolutional Neural Networks (CNN)**:

  CNNs are highly effective for computer vision tasks, including image classification. You can design a convolutional neural network to automatically learn relevant features from images.

- **Transfer Learning with Pretrained Models:**

  By using pre-trained deep learning models on large datasets, you can adapt them to a specific task. Models like VGG, ResNet, or EfficientNet can be beneficial.

- **Recurrent Neural Networks (RNN):**

  If you are working with sequences of images (e.g., dynamic gestures), RNNs or architectures based on Long Short-Term Memory (LSTM) can be useful to capture temporal information in the data.

- **3D Convolutional Neural Networks (CNN3D):**

  If the images include temporal information, such as gestures changing over time, you might consider CNN3D, which extends traditional CNNs into the temporal dimension.

IV. Hypothesis formulation

Due to the lack of resources and computational power, we need a model that is resource-efficient and can operate effectively on less powerful hardware.

In addition, since we are working with images, we require a model capable of efficiently detecting patterns. For this reason, we decided to implement a deep learning model.

Deep Learning is a branch of machine learning that focuses on the use of artificial neural networks to learn and perform complex tasks. It is called "deep" because it involves training models with multiple layers (also known as deep neural networks) to learn hierarchical representations of data. These representations allow the model to capture increasingly abstract features and patterns as it goes deeper into the layers.

With the learning branch clear, we chose to use one found on the internet[4] that served our purpose: was computationally less expensive to train and easier to use.

The dataset format is patterned to match closely with the classic MNIST. Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions). The training data (27,455 cases) and test data (7172 cases) are approximately half the size of the standard MNIST but otherwise similar with a header row of label, pixel1,pixel2….pixel784 which represent a single 28x28 pixel image with grayscale values between 0-255. The original hand gesture image data represented multiple users repeating the gesture against different backgrounds. The Sign Language MNIST data came from greatly extending the small number (1704) of the color images included as not cropped around the hand region of interest. To create new data, an image pipeline was used based on ImageMagick and included cropping to hands-only, gray-scaling, resizing, and then creating at least 50+ variations to enlarge the quantity.

Fig. 1. ASL Alphabet Dataset

Fig. 2. ASL Alphabet Dataset GrayScale

Given that we are dealing with image classification, employing a convolutional neural network (CNN) will yield optimal results, as its strength lies in pattern recognition. By using an appropriate dataset and parameter tuning, we can achieve a very high level of accuracy.

V. Hypothesis Testing through Experimentation

To test the hypothesis, we have used Google Colab, TensorFlow: Keras, supervised Deep Learning, and CNN.

First, we needed 2 MNIST datasets: one for training and another for testing.

Below, you can see the amount of training data:

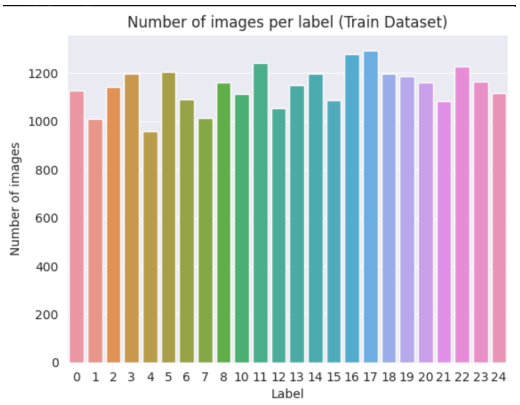

Fig. 3. Amount of train images

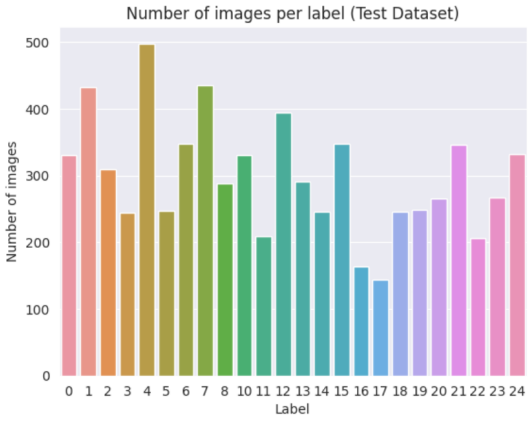

Fig. 4. ASL Amount of test images

We codify the labels with One Hot Encoding for binary representation.

We convert the DataFrames to NumPy arrays. It is necessary to normalize the data and to reshape it from 1-D to 3-D as required by the input of CNNs.

Drawing from the lessons learned in our trial and error processes, we encountered significant overfitting issues. Consequently, we had to incorporate data augmentation to mitigate and prevent such overfitting.

Data Augmentation: To mitigate overfitting issues, it is essential to artificially expand our dataset. The concept involves introducing small transformations to the training data to replicate variations and enhance its diversity.

Approaches that alter the training data in ways that change the array representation while keeping the label the same are known as data augmentation techniques. Some popular augmentations people use are grayscales, horizontal flips, vertical flips, random crops, color jitters, translations, rotations, and much more.

By applying just a couple of these transformations to our training data, we can easily double or triple the number of training examples and create a very robust model.

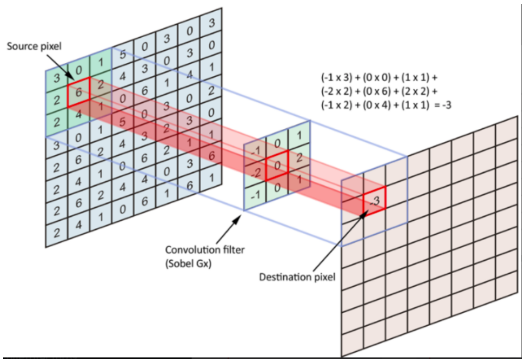

Fig. 5. CNN explanation

Fig. 6.

In this project, we have decided to randomly rotate the images by 10 degrees, shift them vertically and horizontally by 10%, and apply a zoom of 10%. However, we do not apply vertical_flip nor horizontal_flip since it could lead to misclassification.

Now it is time to train the neural network. As proposed in the hypothesis, we use Convolutional Neural Networks (CNN) and 'ADAM'' as the optimizer. The chosen loss function is categorical cross-entropy. We use the Keras function ReduceLROnPlateau, which allows modifying the learning rate dynamically.

The chosen activation function of the last layer is Softmax. It is an activation function commonly used in neural networks, especially in multiclass classification problems. Its main function is to convert the raw outputs of the neural network into probabilities. This is crucial when dealing with problems where input data must be assigned to multiple classes, as is the case here, with 25 classes.

Within the model.fit(), we can incorporate both the training and validation data.

For later visualization and documentation of errors, accuracy, and results, it is necessary to generate graphs and samples of the outcomes

VI. Results Analysis

Based on the experiments we have conducted, we have gathered the following results:

Visualization of learning curves. Accuracy and Loss:

After executing the training and validation of the model, it is time to evaluate the data, and we can observe that the accuracy is 100%.

```
7172/7172 [==============================] - 1s 102us/step
Accuracy of the model is -  100.0 %
```

Fig. 7. Model accuracy

Based on the accuracy obtained from both training and validation, we can observe the following results:

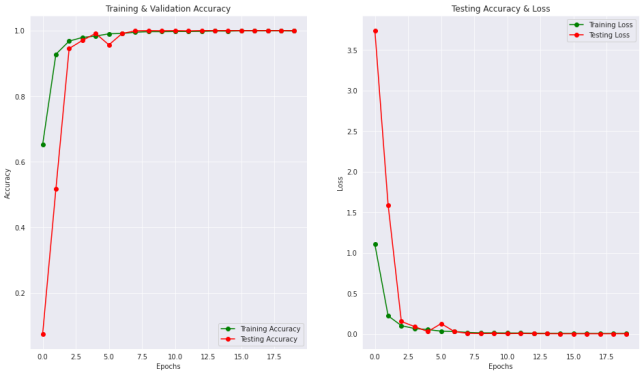

Fig. 8. Learning curves graphs: Accuracy and loss

It shows that immediately, in the second epoch, the accuracy is very high, and by the seventh epoch, it is already perfect. The error is zero due to the 100% accuracy, indicating no overfitting.

Based on predictions with the actual output, we can see that in both cases, the maximum value can be obtained: 1.0.

	precision	recall	f1-score	support
Class 0	1.00	1.00	1.00	331
Class 1	1.00	1.00	1.00	432
Class 2	1.00	1.00	1.00	310
Class 3	1.00	1.00	1.00	245
Class 4	1.00	1.00	1.00	498
Class 5	1.00	1.00	1.00	247
Class 6	1.00	1.00	1.00	348
Class 7	1.00	1.00	1.00	436
Class 8	1.00	1.00	1.00	288
Class 10	1.00	1.00	1.00	331
Class 11	1.00	1.00	1.00	209
Class 12	1.00	1.00	1.00	394
Class 13	1.00	1.00	1.00	291
Class 14	1.00	1.00	1.00	246
Class 15	1.00	1.00	1.00	347
Class 16	1.00	1.00	1.00	164
Class 17	1.00	1.00	1.00	144
Class 18	1.00	1.00	1.00	246
Class 19	1.00	1.00	1.00	248
Class 20	1.00	1.00	1.00	266
Class 21	1.00	1.00	1.00	346
Class 22	1.00	1.00	1.00	206
Class 23	1.00	1.00	1.00	267
Class 24	1.00	1.00	1.00	332
accuracy			1.00	7172
macro avg	1.00	1.00	1.00	7172
weighted avg	1.00	1.00	1.00	7172

Fig. 9. Accuracy of each class

Heatmap:

To obtain the heatmap between the actual and predicted output, it is necessary to use the confusion matrix.

The confusion matrix is a tool used in classification problems to assess the performance of a model by comparing the model's predictions with the actual classes of the data.

We can see that the heatmap aligns with the 100% accuracy since the predicted value matching the desired value has a non-zero value in the position they both seek.

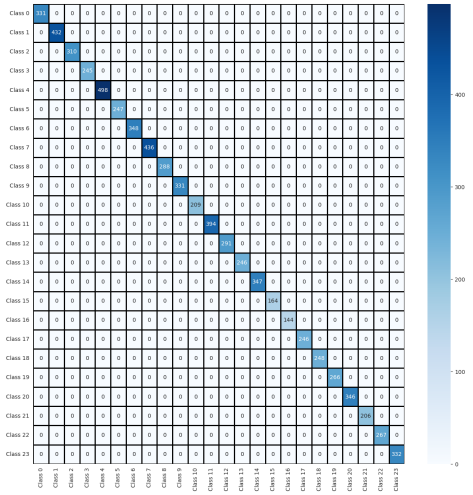

Fig. 10. Heat map

Demonstration of Prediction vs Actual Outcome:

Finally, to facilitate the visualization of the expected output with the actual one, we have demonstrated that the model holds true by showcasing some of the 25 classes; specifically, 6 of them.

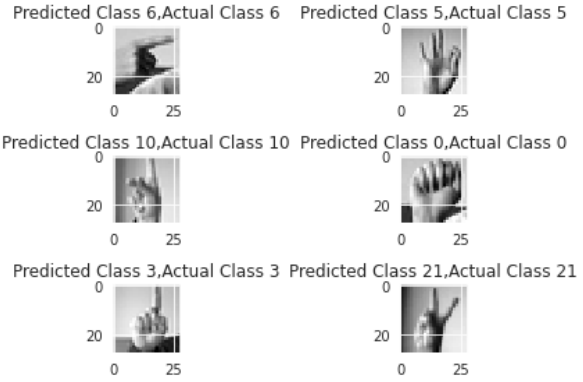

Fig. 11. Visual demonstration of predicted output and actual output

VII. Conclusion

After conducting all the tests and compiling all the results, it can be deduced this is an effective hand sign recognition method.

In this case, for the recognition using a dataset, where the main problem is to classify different signs.

As it is explained, the accuracy obtained is 100%, avoiding at the same time the overfitting and false no error situations, expanding the datasets virtually

In a social environment, the group's involvement in this project had the objective of integrating the population sector with the disability of deafness.

In order to include future improvements, it would be captivating to apply this recognition system in a real time situation, using cameras, different illumination situations and using full-body pictures. Moreover, the *J* and *Z* signs, because their signs are dynamic signs, thus can not be processed as a single image. The final goal could be the creation of a software which could detect whole phrases, unifying previous hand sign recognitions.

References

[1] SpreadTheSign, "Sign Language dictionary,". Available in: Diccionario de lengua de signos | SpreadTheSign. Accessed on: 20/12.

[2] L. Marie, "Sign language alphabet recognizer using Python, openCV, and TensorFlow," GitHub.. Available in: loicmarie/sign-language-alphabet-recognizer: Simple sign language alphabet recognizer using Python, openCV and tensorflow for training Inception model (CNN classifier). (github.com). Accessed on: 20/12.

[3] Kaggle, "[EDA] ASLFR - Animated visualization,".. Available in: [EDA] 🤟ASLFR🖖 - Animated visualization📊🧍 | Kaggle. Accessed on: 20/12.

[4] Hugging Face, "AI_ML repository,"]. Available in: Kolpitor/AI_ML at main (huggingface.co). Accessed on: 20/12.