Final report ”Digital Whiteboard”

Adrian Flores Camacho
Andrés Saúl Bellido Gonzales
Alejandro Núñez Arroyo
Raul Angel Mollocuaquira Capariconа
Universidad Católica Boliviana “San Pablo”
La Paz, Bolivia

***Abstract*—The present work consists of the implementation of a hand recognition algorithm for tracing letters and numbers in a space visible to a computer camera in order to obtain a ”virtual whiteboard” and achieve a better explanation by a teacher or speaker in a video conference who needs to use a whiteboard and point with gestures while explaining the content of the ideas to be shared.**

***Index Terms*—Computer Vision, Handwriting, Recognition, Letters, Numbers.**

I. Introduction

Virtual classes and videoconferencing have become a daily topic in recent months, largely due to the emergence of the global pandemic that forced us to have a new methodology of study. The virtual classes were of great help, but there are some aspects that hindered the explanations in class, for example the tracing of numerical characters and letters, teachers and exhibitors use the mouse to point or write on the screen, they do not have an electronic device such as a tablet with which they can write these characters, also the handwriting of each person is different, these characteristics complicate the proper understanding of what is to be transmitted. The algorithm shown in this paper provides a solution to these problems.

II. Related Work

A. Offline Handwritten Text Recognition

Offline handwriting recognition involves the automatic conversion of text in an image into letter codes that are usable within computer and text-processing applications. In simple terms, it is the text extraction from your handwritten notebooks/pages. Why called offline? The point being that there is an online text recognition system, which is referred for text that is digitally generated by using tools like stylus, apple pencil, etc. [1]

B. Position-Free Hand Gesture Recognition Using Single Shot MultiBox Detector Based Neural Network

Tang et al. proposes a hand gesture recognition system that can detect and determine a hand gesture in any position of the screen. The Single-Shot MultiBox Detector (SSD) is meant to find the hand’s position and Convolutional Neural Network (CNN) to classify a hand gesture. Real-time video feed is captured with a web camera, the image is transferred to a trained SSD network to produce a bounding box on the hand that has the best score of representing a hand. A frame along the bounding box is cropped to produce a segmentation and passed into CNN, thereafter a gesture is predicted. [2]

III. Data

We looked for a different dataset to MNIST which was the basis for presenting the results in Project milestone [3], we looked for the new dataset to allow us not only to have the information of numbers between 0 and 9, but also to contain different numeric characters and letters. For this we will use the implementation of the Handwritten math symbols (CROHME dataset), which has different classes of characters with more than 100,000 samples. [4]

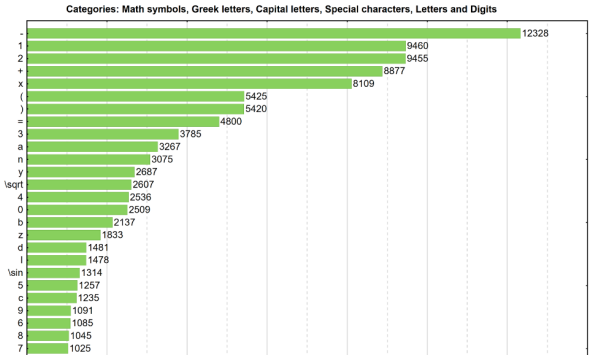

Fig. 1: Categories of the Handwritten math symbols (CROHME dataset dataset).

A readjustment of all the images of the dataset was made, the same ones have a dimension of 45x45 pixels, in addition that these did not have a border so that the content of the image is in the center, for it a readjustment of dimension was applied to 28x28, a border to the image and also the function bitwise to invert the colors.

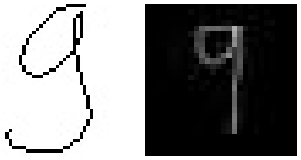

Fig. 2: Original image to the left, modified image to the right

The same process was performed for the following data, which are considered for character detection in the project deliverable:

- Numbers: 0 to 9.
- Numeric characters: +, -, *, /, =.
- Letters: a, c.

Once this process has been completed for all the images, we will proceed with the training in our CNN architecture.

IV. Methods

To carry out this project, different methods were used, ranging from adding value to it and improving the recognition of the numbers and characters in the images.

A. Libraries

1) OpenCV: OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.
As seen throughout the course, this library is very useful for working with images, in this case, it was used for reading, rotating, changing the color space, binarized, and operators, or operators, resizing, and concatenation. of pictures.

2) Mediapipe: MediaPipe, an open-source framework designed specifically for complex perception pipelines leveraging accelerated inference (e.g., GPU or CPU), already offers fast and accurate, yet separate, solutions for these tasks. Combining them all in real-time into a semantically consistent end-to-end solution is a uniquely difficult problem requiring simultaneous inference of multiple, dependent neural networks. [5]

Mediapipe was very useful to us to add value to our project, mediapipe has functions for face recognition, face meshing, object detection, face and hair segmentation, etc. The function that interested us for the project was the detection of hands, which allowed us to detect a hand next to the characteristic points, a function included in mediapipe is to obtain the coordinates of these points that helped us to verify if a finger is raised and if it is not, and this was very useful to perform drawing functions in the interface.

3) Torchvision: Torchvision provides access to datasets, models and prepossessing facilities for deep learning with images. Integrates seamlessly with the 'torch' package and it's 'API' borrows heavily from 'PyTorch' vision package. [6]

V. Experiments

1) Train: The proposed architecture for classifying the data classes was based on the work done by Amitrajit Bose with the Handwritten Digit Recognition project [7], in the same way we used part of the code seen in Tutorial 16: Neural Networks for Image Classification (Part 2). [8] The main difference between those codes and the one presented in this work is that the number of classes is different, being 10 for the mentioned works and 17 for ours, in addition to obtaining data from the previous works we used the function datasets.MNIST that allows downloading data in a different format than jpg, png or others, in the case of the project we made this modification to load the image data separated into a train and test value for each category, in the future this may help to have many more classes than those presented in this work.

For the classification of the images a CNN was used, the layers can be seen in the following image.

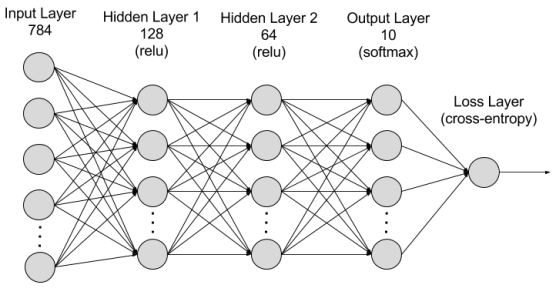

Fig. 3: Neural network architecture. [9]

To build our network we will use the figure above, having an input layer of 784 since our data are binarized images of 28x28 pixels and our output layer is equal to our class number 17. The output layer is a linear layer with LOGSOFTMAX activation generally used in sorting tasks, we use the torch library's own function to apply this layer.

$$\text{LogSoftmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right)$$

Fig. 4: PyTorch LOGSOFTMAX function. [10]

2) Model interface and prediction: The code begins with the import of the libraries used. A dictionary was declared with the values of the classes of the trained model, the already trained model is also loaded with the extension ".pt" and some variables to facilitate coding.The images were read to make the program interface and a Video capture and an array with values of 0 were declared as a canvas image for the drawing.

A while loop was created to obtain frames from a camera and almost all the algorithm within it, the detection of hands as mentioned in a previous section was done with mediapipe, which obtains the most characteristic points of a hand, and the coordinates of those points, from the distances between the coordinates it was verified if the finger of the identified hand is extended, and as a result, it gives us a vector with values of 0 or 1 for each finger, this helps us at the time of creating functions for the program.

There is the selection mode when only the index finger is extended, in this mode, you can select between 4 colors,

Fig. 5: Hand detection and segmentation.

with RGB values of (255,0,0), (0,255,0), (0,0,255), and (0, 0,0), in the drawing mode that is entered with the index and middle fingers extended, with the colors and the ”cv2.line ()” function you can draw lines on the created canvas and on the same image obtained by the frames in the coordinates of the tip of the index finger.The last drawing function you have is to clean all the drawings that can be activated if only the pinkie finger is extended.

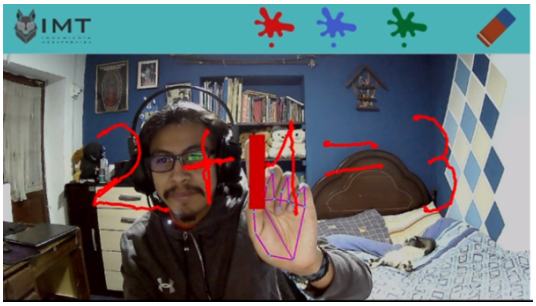

Fig. 6: Virtual whiteboard interface

From the image of the canvas, a transformation of the color channel from RGB to grayscale is carried out, to then perform a binarization of the image, as a resulting image the following is obtained:

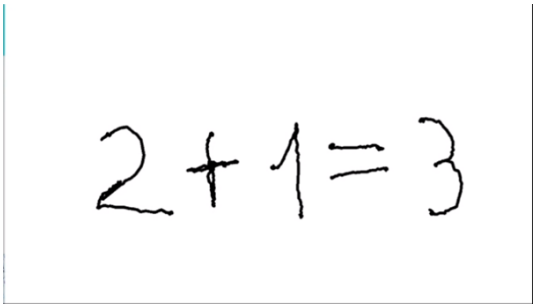

Fig. 7: Binarized Canvas Image

Once this function is established, the drawings must be recognized by the trained model, so by pressing the ”f” key the program takes the binarized image as input, but before entering it into the model, the image is processed with a mask to recognize the black color in the image, and with the ”FindContours” function of OpenCV the edges are detected and a rectangle is generated for each contour generated, using a dictionary the characters of the image are ordered from left to right, they are resized to a 28x28 pixel image.

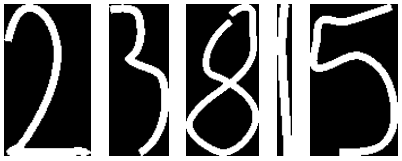

Fig. 8: Cutting of each number in the image

These images are read and recognized by a trained model, which defines the class to which the trace belongs, outputting the predicted character.

Finally, this value predicted by the model is taken as the index of the dictionary previously generated with the classes, and finally, all these values are concatenated to be able to be put on another canvas but as text, the result is the following:

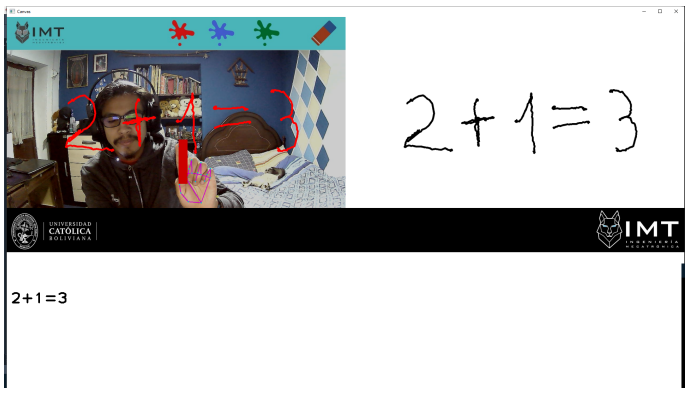

Fig. 9: User interface

3) Metrics: In this section, we carry out an evaluation of the numbers and signs model using the metrics already acquired in the classes, which will show the Iterations vs Accuracy table, the CNN classification report table and the confusion matrix of the labels and predictions.

The following figure shows the Iterations vs Precision which shows us how the accuracy increases when doing more iterations during the training, managing to observe that the precision little by little increases its accuracy to 100

In this section you can see the table of the metrics that shows us the precision of each class that we observe that has a value greater than 0.75, the values are close to 1 showing high precision in most classes and then in the The next row shows a recall that shows us the false negatives that each class presents, where it can be observed that most of the

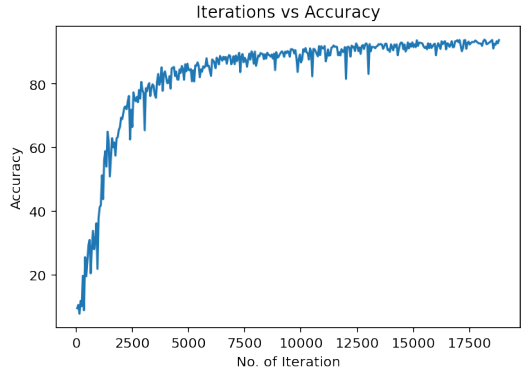

Fig. 10: Iterations vs Accuracy

classes have a value close to one, which shows that there are not many false negatives in the classes. The metric f1 score shows us the precision and a perfect recovery, that if the value is close, there is good precision and a perfect recovery, which we observe in our figure that most of our classes have a value above 0.76 and some values above 0.9 so it gives us good precision and recovery in the classes. Finally, we observe that the support metric shows us the number of images that are true responses for each model, showing that each class has more than 207,000 true images for each class.

Classification report for CNN :

	precision	recall	f1-score	support
0	0.90	0.91	0.90	208419
1	0.79	0.80	0.80	207652
2	0.86	0.81	0.84	207973
3	0.84	0.83	0.84	208249
4	0.75	0.77	0.76	208148
5	0.87	0.81	0.84	208678
6	0.89	0.92	0.91	208033
7	0.91	0.87	0.89	207252
8	0.90	0.92	0.91	208275
9	0.87	0.88	0.88	208179
10	0.81	0.80	0.80	208051
11	0.98	0.99	0.99	208034
12	0.88	0.88	0.88	207798
13	0.94	0.95	0.95	207386
14	0.87	0.92	0.89	208424
15	0.87	0.86	0.86	207796
16	0.93	0.94	0.93	208403
accuracy			0.87	3536750
macro avg	0.87	0.87	0.87	3536750
weighted avg	0.87	0.87	0.87	3536750

Fig. 11: Metrics Classification Report.

The following image shows the confusion matrix that shows the behavior of the data in the model that shows the performance of the algorithm that shows that the diagonals of the confusion matrix all perform well but the surrounding data shows the data that is confused in The classes that show the intensity of the color of the percentage of false positives and negatives.

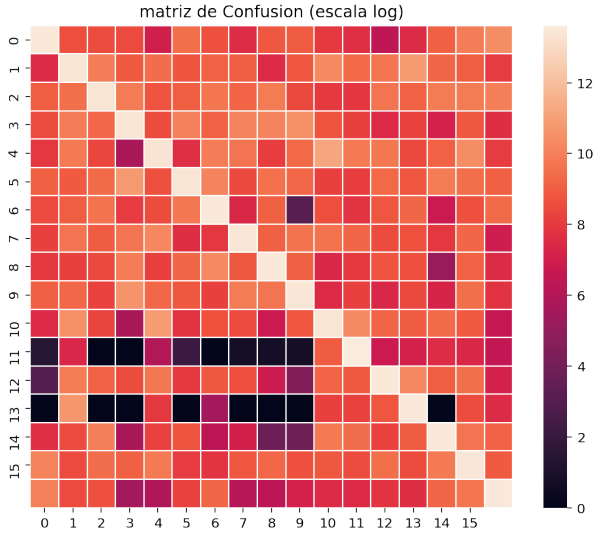

Fig. 12: Confusion Matrix

VI. Conclusion

In conclusion, the main objectives were met, which are the recognition of hand gestures through the mediapipe library, which thanks to the script allows us to recognize all the actions we perform on the virtual whiteboard in which, through the algorithm of the torch, it is possible to cut and process the numbers, signs and letters a and c of what the virtual whiteboard shows, recognizing the texts written in real time, obtaining the characters in a clean format which to achieve this recognition we had to make a .pt model with the characters we need for the virtual whiteboard which was a challenge in the realization of the model since there was similarity between some characters which was chosen to increase the epoch and select the characters well so that there is no confusion in the strokes made by the user, achieving good results when applying the required changes, achieving a good recognition of what s characters drawn on the virtual whiteboard, obtaining a recognition of each number on the canvas to show us in a clean way the strokes made by the user.

VII. Supplementary Material

https://github.com/AlejandroNunezArroyo/ComputerVision-Project

https://www.youtube.com/watch?v=gNZLOuidODo

References

[1] . Anofryev (2021, May). Offline Handwritten Text Recognition [Online]. Available:https://www.kaggle.com/alexeyanofryev/offline-handwritten-text-ocr/execution

[2] . Z. Ren, J. Meng, J. Yuan, and Z. Zhang, (2011) Position-Free Hand Gesture Recognition Using Single ShotMultiBox Detector Based Neural Network [Online]. Available: https://doi.org/10.1145/2072298.2072443.

[3] . (2021, Apr). THE MNIST DATABASE of handwritten digits [Online]. Available: http://yann.lecun.com/exdb/mnist/

[4] . (2021, Apr). Handwritten math symbols (CROHME dataset dataset) [Online]. Available: https://www.kaggle.com/xainano/handwrittenmathsymbols

[5] . OpenCV team (2021, Apr). Open Source Computer Vision Library [Online]. Available: https://opencv.org/about/
[6] . Grishchenko I., Bazarevsky V. (2020, Dec). MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device [Online]. Available: https://ai.googleblog.com/2020/12/mediapipe-holistic-simultaneous-face.html
[7] Torch Contributors (2021) TORCHVISION [Online]. Available: https://pytorch.org/vision/stable/index.html
[8] . (2021, Apr). handwritten-digit-recognition [Online]. Available: https://github.com/amitrajitbose/handwritten-digit-recognition
[9] (2021, Apr). [Online] Available: https://colab.research.google.com/drive/1E8skjsRJRUfZHzPTQKVpG3UAiUgWNt7
[10] (2021, Apr). [Online] Available: https://pytorch.org/docs/stable/generated/torch.nn.LogSoftmax.htmltorch.nn.LogSoftmax