INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Challenge: Digits Recognition

Course: Artificial Intelligence

Team:

Carlos Roberto Cueto Zumaya	A01209474
Alan Kuri García	A01204805

Professor:

Ruben Stranders

Due Date:

November 20, 2018

Our goal was that anyone with a smartphone could digitize a document with just one photograph, however, we had two limitations, time and complexity, the first was to short and achieving something of such magnitude was not possible so we decided to narrow the project to just recognize digits as this would serve as the beginning and at some point continue with the original idea.

Existing solutions for the digitization of information are a point of interest for society, continuously large amounts of written information are generated, in that context, and with a world connected to the internet, it’s important that all people have access to it, automating the process implies significant savings in human resources and an increase in productivity.

OCR (Optical Character Recognition) is a process focused on the digitization of texts it extracts text and numbers from images, storing in some form or type of data and then interact with them in a text editor. The process consists of four stages:

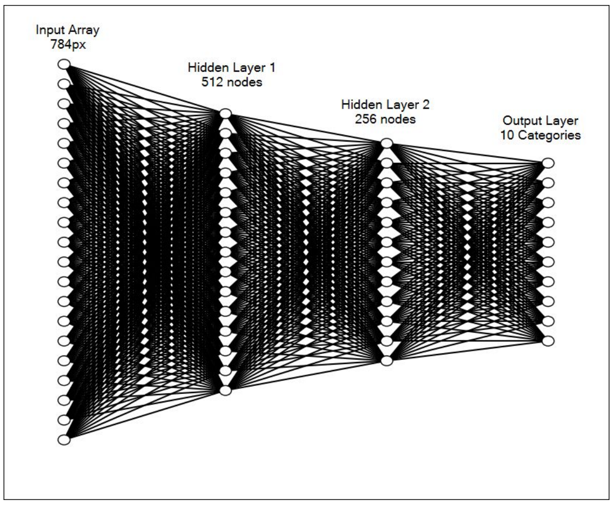
- Binarization: The image is converted to only two colors, black and white.
- Fragmentation: One of the most expensive steps, involves the segmentation of the image by decomposing the text by its lines and strokes.
- Flattening: Once the character is isolated, the image is converted into a binary matrix and using the formula of absolute distance, a circle is created dividing the matrix/image into different quadrants, comparing each of them produces a probability of being N character.
- Comparison: Usually to have a better approximation, the image is compared with previously selected templates and a result is given.

However, the success depends on many factors, to begin with, on the nature of the document to be digitized, print quality, paper wear, print resolution, color and type of ink, typography and readability are just some elements that prevent good detection.

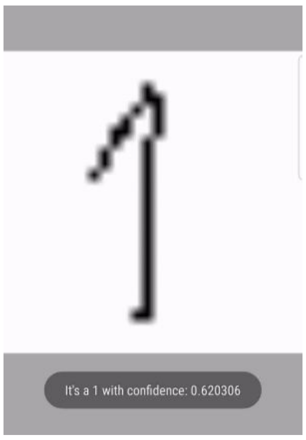
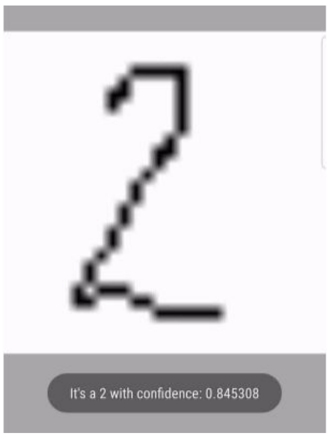
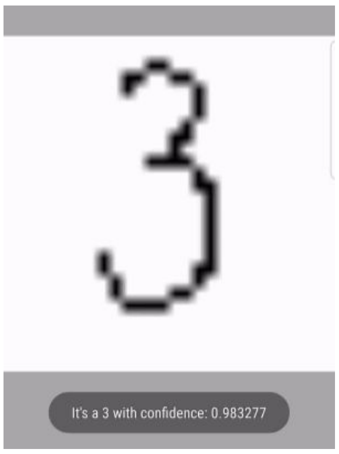
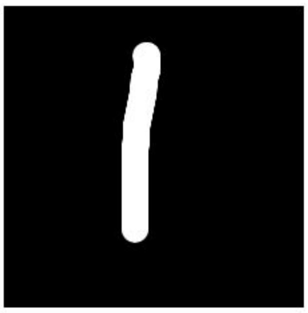
Our solution consists of a CNN (Convolutional Neural Network), a special type of neural network whose computational power to process images is impressive, usually they can be divided in two parts: a feature learning and a classification part; we achieved the first one by combining one convolution layer and a pooling layer, these are the Hidden Layers 512 and 256 nodes respectively, two Dropout layers are added just as a regularization layer to randomly exclude 20% of nodes to prevent overfit and finally we achieved the classification by dense layer as an output layer.

The dataset was constructed from images taken from a variety of scanned documents, normalized in size and centered, each image is a 28 x 28-pixel square (784 pixels total), since it’s a classification task and there are 10 digits we have 0 to 9 classes to predict so we came up with the next network architecture:

- Input Layer: 784 nodes
- Hidden Layer 1: 2^9 nodes
- Dropout: 0.2
- Hidden Layer 1: 64 nodes
- Dropout: 0.2
- Output Layer: 10 nodes

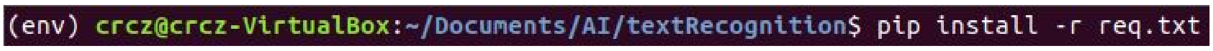


We didn't found any tool that provides us with as much data as we wanted and the tutorials over internet followed exactly the guide of TensorFlow's documentation, however, there is an app developed for android that has an implementation based on TensorFlow with a different network architecture. Below we show the comparison of some digits and its confidence value, where we can see that both have a good similar performance and the success rate.

 It's a 1 with confidence: 0.620306 Confidence: 0.620306	 It's a 2 with confidence: 0.845308 Confidence: 0.845308	 It's a 3 with confidence: 0.983277 Confidence: 0.923272
 Confidence: 0.54059898853302	Confidence: 0.999771118164062	Confidence: 1

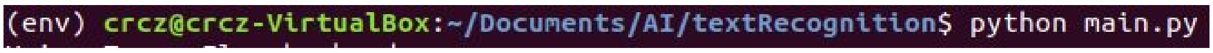
Setup

We used several libraries that facilitated the CNN creation, graphicss and operations over matrices, we created a requirements file which contains all the packages so that it could be executed without any problem, to install them you need to type *pip install -r req.txt* on the root folder of our repository.

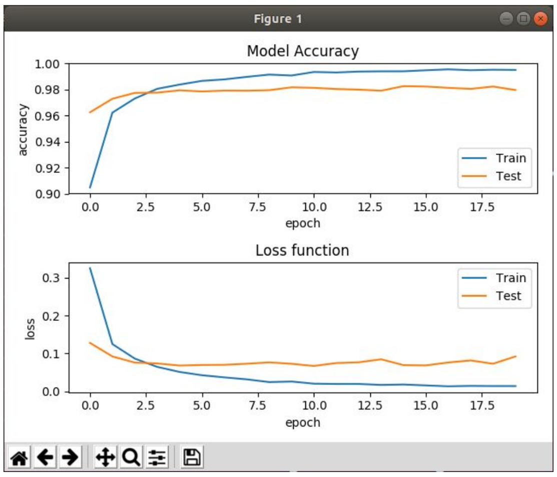


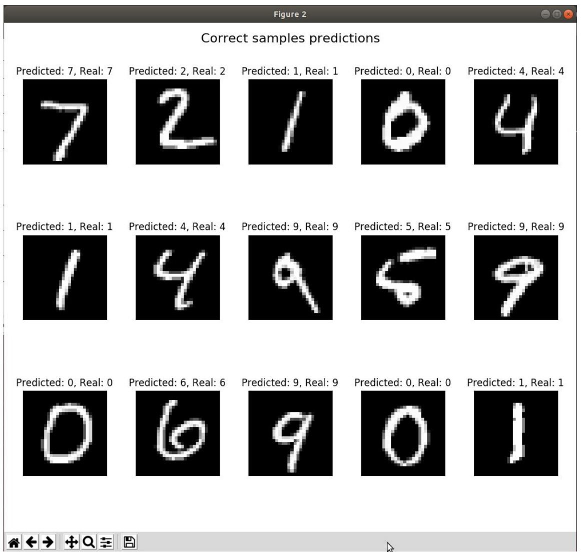
```
(env) crcz@crcz-VirtualBox:~/Documents/AI/textRecognition$ pip install -r req.txt
```

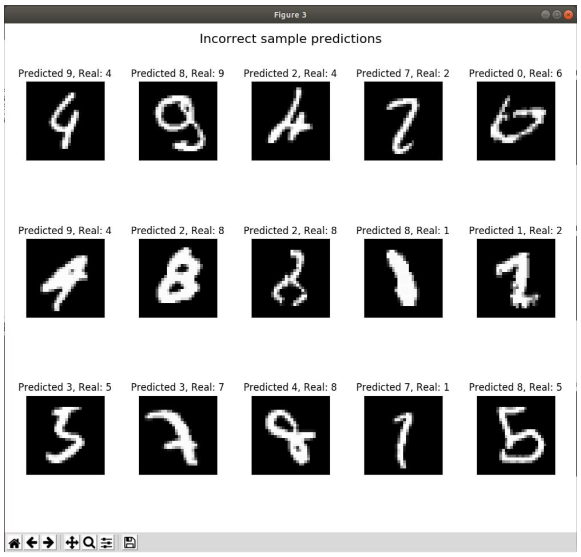
Type *python main.py*, to execute the program, the dataset will be downloaded, the CNN model will be created, trained and tested, the performance is calculated, and three windows will show up with 15 samples correctly predicted and 15 samples incorrectly predicted, and two graphics that show the model accuracy and its loss function.



```
(env) crcz@crcz-VirtualBox:~/Documents/AI/textRecognition$ python main.py
```

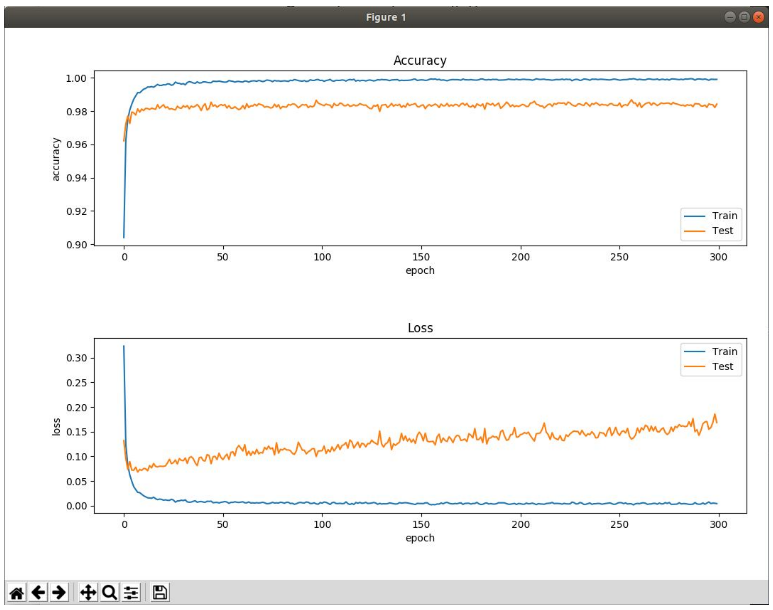






Analysis

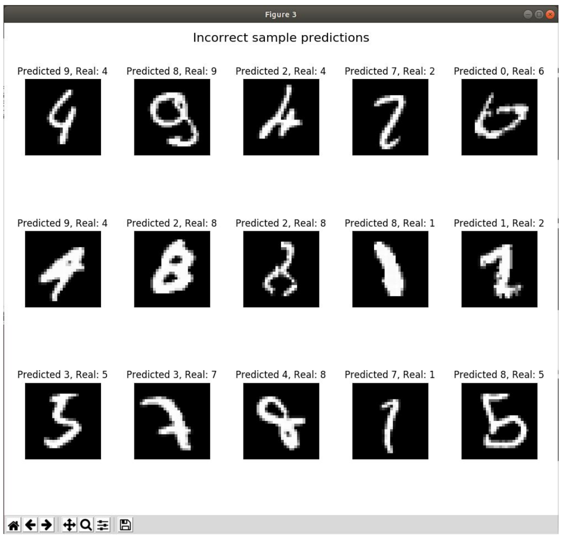
The Loss function seems to have an abnormal behavior from 0 to ~2.5 epochs, the first problem we discovered was that we repeated 10,000 images on cross-validation and test set so we changed the data to 50,000 test images, 10,000 cross-validation images and 10,000 test images making sure that the last ones have never seen before, but nothing changed and with more epochs the CNN overfit with time, as you can see on the image below. To prevent overfit we decrease the epochs and added two dropout layers but it was still the same.

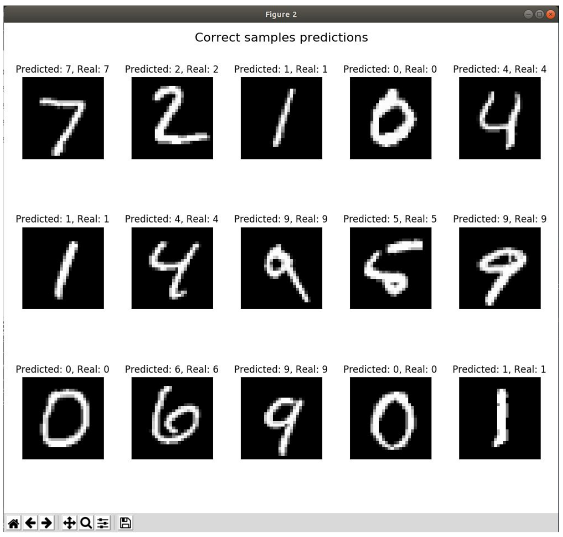


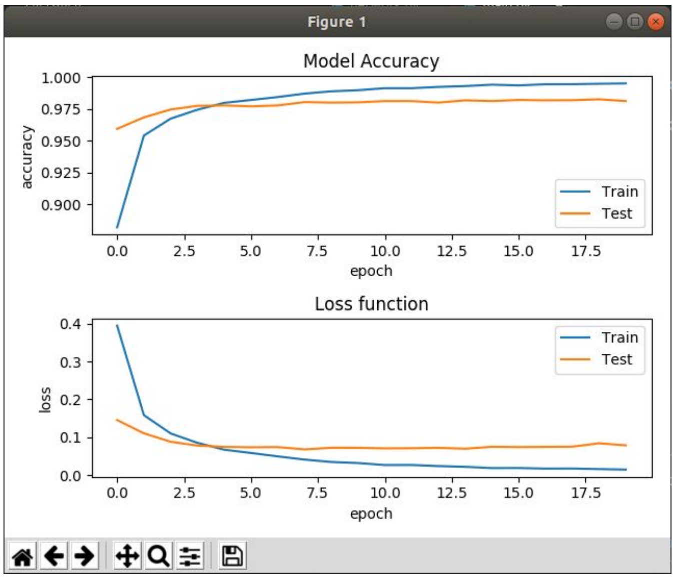
What we think is happening is that the data between training and validation could have a couple of samples duplicated and maybe we needed to reevaluate our splitting method, add more data or change the performance metric, but without time to going further into the tutorials the only idea that sound coherent to us is that maybe the training set is more difficult than the test set or we may misunderstood the values and the loss value is high because the CNN knows nothing of the training set but for the test set as it already have knowledge the loss value is lower.

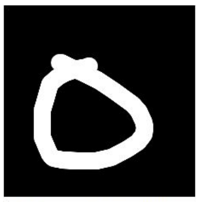
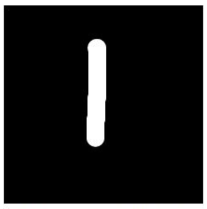
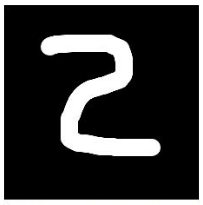
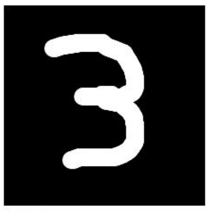
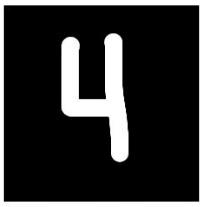
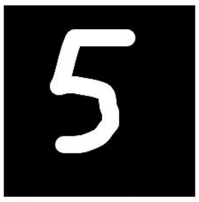
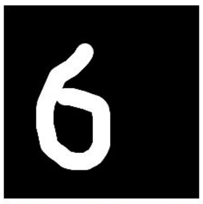
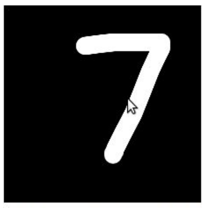
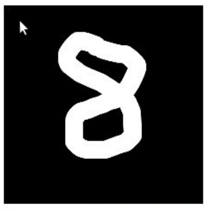
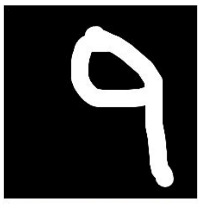
The base parameters we tested were simple, we have 784 px so 784 nodes as input layer and 10 numbers so 10 nodes on every layer the results were quite good but not enough, we changed the number of nodes, the epochs and the batch size until we got a good error rate, below we show the most significant tests that we obtained.

Test	Dimenssion	Epoch	Batch Size	Correct	Incorrect	Accuracy	Error	Loss
1	784,10,10,10	10	100	9129	871	91.29%	8.71%	0.30
2	784,10,10,10	40	250	9251	749	92.51%	7.49%	0.26
4	784,32,32,10	20	200	9592	402	95.98%	4.02%	0.13
5	784,512,64,10	20	200	9801	199	98.01%	1.99%	0.095
6	784,512,256,10	20	200	9821	179	98.21%	1.79%	0.078







 Confidence: 0.949326097965	 Confidence: 0.92642807960	 Confidence: 0.852109789846	 Confidence: 1	 Confidence: 0.999831199645
 Confidence: 0.999514698982	 Confidence: 0.920389592647	 Confidence: 0.887199699878	 Confidence: 0.942253768444	 Confidence: 0.99975401163

On our tests using the web application as a digit tester, we faced problems when resizing the image and center the number, we achieved the first one but not the second one, that being said, we believe that we achieved a good project with a success rate close to 100% and the wrong predictions are quite forgivable since in some cases even we couldn’t recognize the digits. On a classification problem real world performance does not really depend on success of the error rate, it is crucial to monitor the model performance and periodically adjust the model to keep its high accuracy.

The project turned out to be a more complicated task than we thought, the improvements that we suggest is change the splitting and shuffling method when creating the cross-validation set and isolate the number of the canvas when using the web tester, so the behaviour of the graph is the correct.

Referencias

3Blue1Brown (2017, October 5). But What *Is* a Neural Network? | Deep Learning, Chapter 1. Retrieved from YouTube: www.youtube.com/watch?v=aircAruvnKk&t=1011s.

3Blue1Brown (2017, October 16). Gradient Descent, How Neural Networks Learn | Deep Learning, Chapter 2. Retrieved from YouTube: https://www.youtube.com/watch?v=IHZwWFHWa-w&t=2s

Ali, S., Sumari, P., Al-Taweel, S., & Husain, A. (2009). *Semantic Scholar.* Retrieved from Digital Recognition using Neural Network: https://pdfs.semanticscholar.org/e248/4f2b2d53f3d53395e5ab79be15adf867cb29.pdf

Geitgrey, A. (2016, June 13). *Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks.* Retrieved from Medium: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

Karn, U. (2016, August 11). *An Intuitive Explanation of Convolutional Neural Networks.* Retrieved from The data science blog: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Kraszewski, B. (2018). Retrieved from MachineLearningDemo: https://github.com/bkraszewski/MachineLearningDemo

Loots, M., Camarzan, D., & Witten, I. (2006). *Greenstone Digital Library Software.* Retrieved from OCR: Reconocimiento Óptico de caracteres: http://www.greenstone.org/manuals/gsdl2/es/html/Chapter_ocr.htm

Murty, N. (2011). *An application: Handwritten Digit Recognition.* Retrieved from https://link.springer.com/chapter/10.1007/978-0-85729-495-1_11

Miyoshi, K. (2016). Retrieved from Tensorflow MNIST demo on Android: https://github.com/miyosuda/TensorFlowAndroidMNIST

Rosner, F. (2018, May 28). *Handwritten Digit Recognition Using Convolutional Neural Networks.* Retrieved from https://dev.to/frosnerd/handwritten-digit-recognition-using-convolutional-neural-networks-11g0

Socorro, S., Morales, G., & Pavel, C. (2012). *Revistas UNAM.* Retrieved from Los problemas de identificación de caracteres OCR para la recuperación de texto en un libro antiguo: http://www.revistas.unam.mx/index.php/rbu/article/view/32557/29901

Sharda, Aryaman. How Does Optical Character Recognition (OCR) Work?“ *YouTube,* YouTube, 20 Nov. 2017, https://www.youtube.com/watch?v=cAkklvGE5io

Tankala, A. (2018, May 26). *Handwritten Digit Prediction using Convolutional Neural Networks in TensorFlow with Keras and Live Example using TensorFlow.js.* Retrieved from Medium: https://medium.com/coinmonks/handwritten-digit-prediction-using-convolutional-neural-networks-in-tensorflow-with-keras-and-live-5ebddf46dc8

Woodford, C. (2018). Retrieved from Optical character recognition (OCR): https://www.explainthatstuff.com/how-ocr-works.html