Tec de Monterrey  Campus Querétaro

Arturo Rivera Flores - A01202457
Cuauhtemoc Suarez - A01206503
Francisco Villanueva - A01202413

Artificial Intelligence
08/25/16

# Report Connect 4 with Digit Recognition. Part 2: Digit Recognition

As a part of the connect-4, the way of introducing the commands without using the command line. At the beginning, it was thought using speech recognition for the commands but the problem with these is that speech recognition requires Bidirectional Recurrent Neural Networks (RNNs) and LTSM which are modules of Long Term Short Memory (Gibiansky, 2014), which is pretty difficult to implement. Thus, digit recognition was decided to be implemented via drawing on a blackboard and detecting the desired column where you want to move.

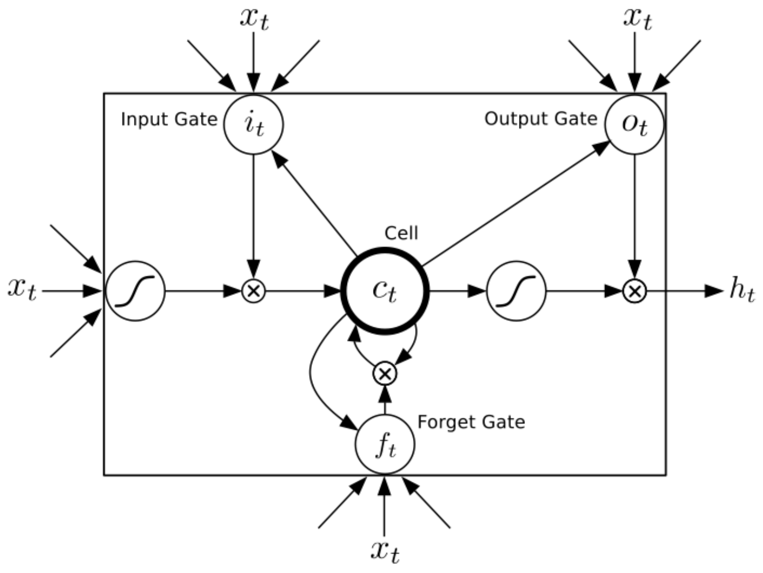


Figure 1

Deciding to implement the digit recognizer was also based in the fact that there is a database called MNIST where there are 60,000 28x28 pixels images of several drawn digits by different people in the United States (Lecun, 1998). P.D. The code and files used for the process of reading this data was taken from the book Neural Networks and Deep Learning of Michael Nielsen (2016).

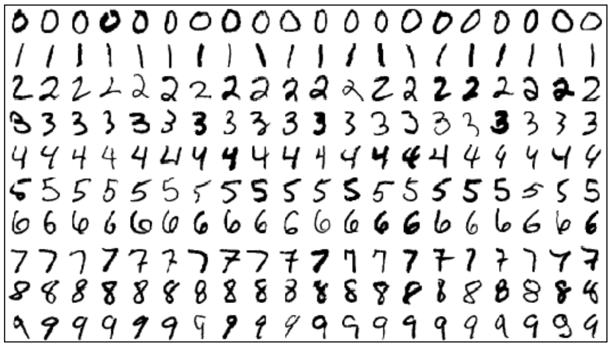


Figure 2 Example Images of MNIST

## Justification

As it was mentioned, this was chosen for giving instructions to the Connect-4. As we are trying to detect hand-written digits, the number of ways a person can draw the same number is too high so an AI is needed for recognizing this numbers even when it is drawn in different ways, it needs to learn from different images and gather the essential characteristics of each number so it can detect further drawings.

There are many different solutions for solving the digit recognition problem; there are 3 in special which uses different methods: random forests, support vector machines, and neural networks. Michael Nielsen (2016) used the last two approaches for solving the problem; the results for the accuracies are shown in the next figure.

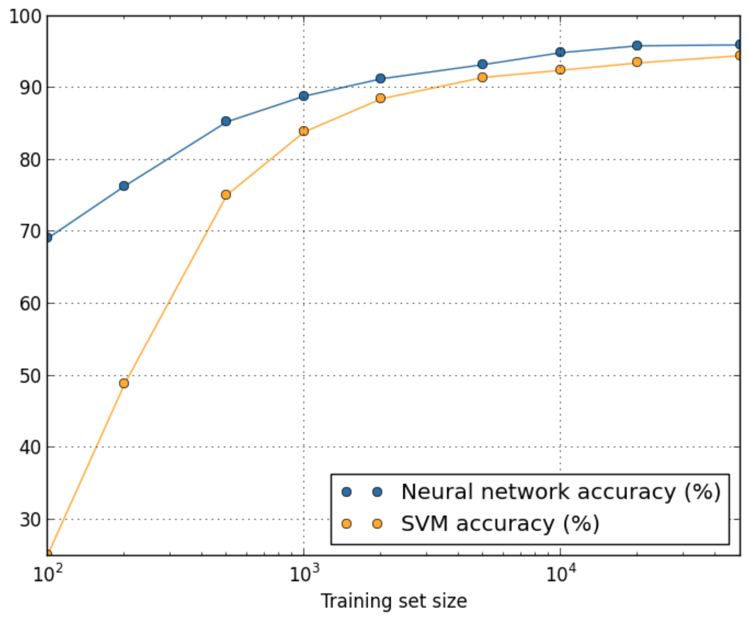


Figure 3 Accuracies comparison for SVM and Neural Networks.

Moreover, Simon Bernard (2007), applied different Random Forests for solving the digit recognition problem, the different implementations vary according to the L number of trees; the results were:

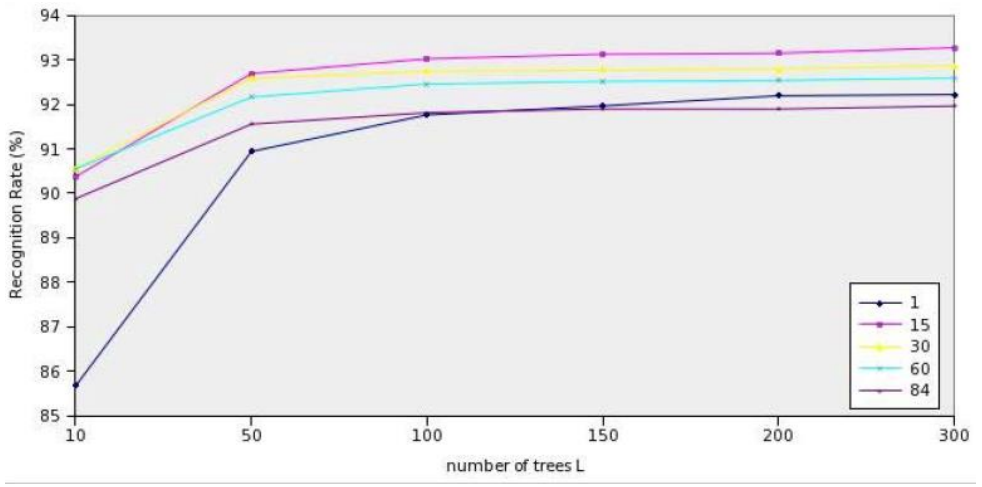


Figure 4. Accuracies with different random forests

Comparing the results obtained by these three methods, the Neural Networks stands tall with a ~95% of accuracy over the ~94% of the SVM and the ~93% of the Random Forests. Thus, the neural network approach seems to be better.

Once the method was chosen, using a Neural Network with only one hidden layer was the easiest step for avoiding overfitting and only having to select the number of neurons for the layer. The algorithm is based in the one described by Michael Nielsen (2016); this is based on using the MNIST dataset for training and testing the Neural Network (50,000 images for the training and 10,000 for the testing).

Overall, the structure for this ANN is having 748 input neurons (for putting the grayscale value 0.0-1.0 of the 28x28 pixels image), a variable number of hidden neurons, and 10 output neurons one per each digit, so the predicted digit is obtained by gathering the one with the highest value in the output… Lastly, the activation function is a sigmoid so this output is considered a likelihood of being one digit or another. The graphical structure is:

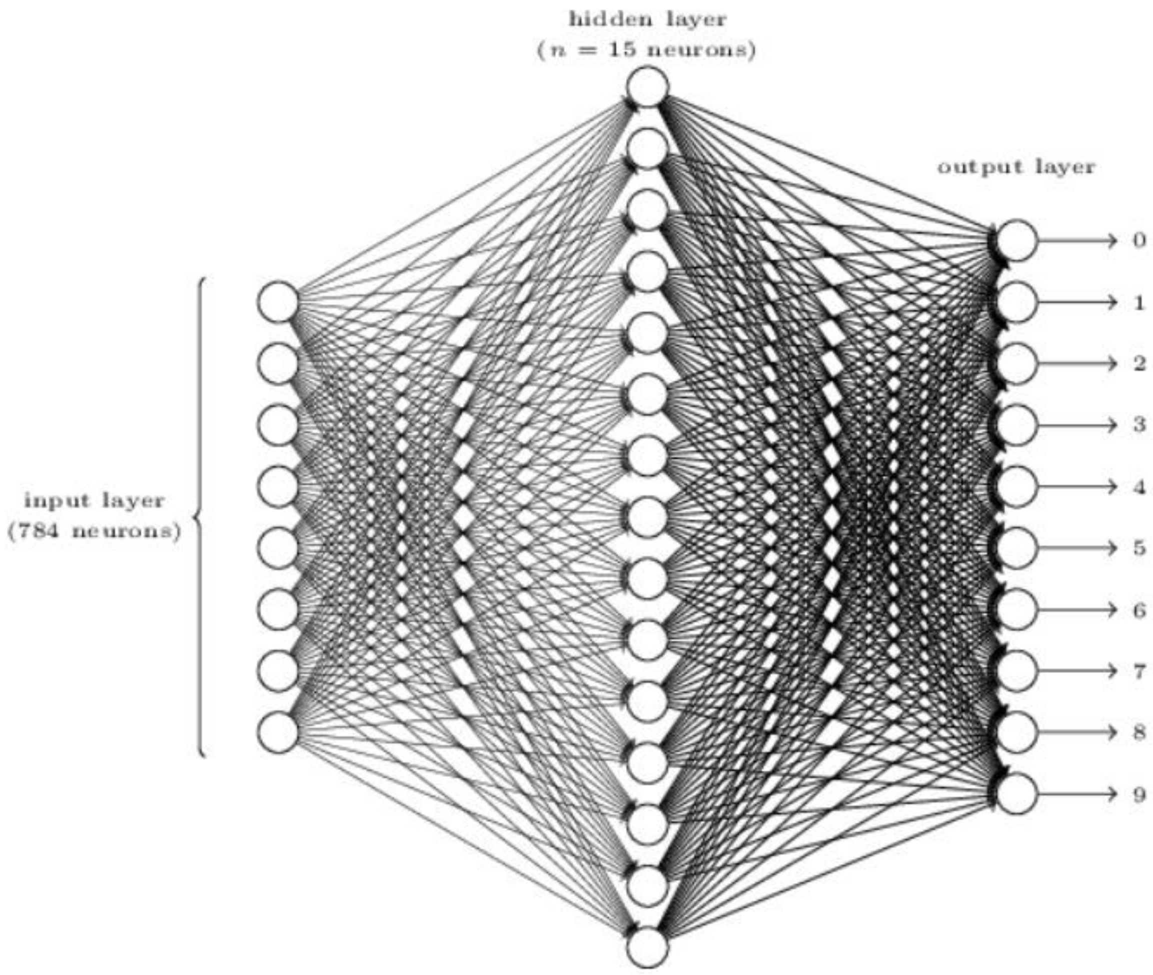
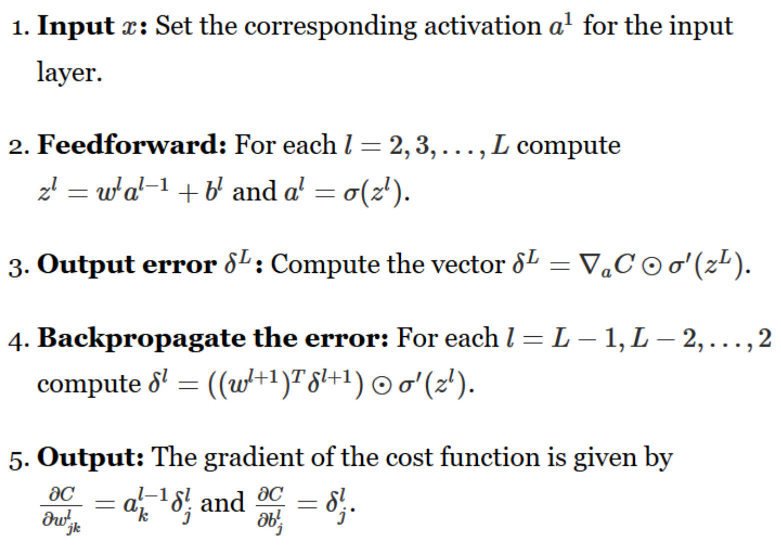


Figure 5. Structure of the  ANN

The complete algorithm is based on a Stochastic Gradient Descent using back propagation, the algorithm of this part is as follows:



1. **Input $x$:** Set the corresponding activation $a^1$ for the input layer.
2. **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.
3. **Output error $\delta^L$:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.
4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.
5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

Figure 6. Back propagation algorithm

Furthermore, this algorithm can be greatly improved in the initialization of the weights and biases as it is done randomly without really controlling where this value can be. For further understanding of the algorithm, visit: http://neuralnetworksanddeeplearning.com

Lastly, for communicating the ANN to the real world, an Open CV program was used where the program detects and frames a written digit and then the image is resized to 28x28 pixels, then it is converted to a column vector with the grayscale values, 0.0 for white and 1.0 for black. This is sent to the ANN and the output of this one is sent to the Connect 4 algorithm so the player can take his move.

**Analysis**

As the implementation of the ANN was fixed to only selecting the amount of neurons in the hidden layer, the number of epochs for the training, and the learning rate, the following table was obtained for several combinations of values:

| Number | Neurons | Epochs (iterations) | Learning Rate | Accuracy |
|---|---|---|---|---|
| 1 | 20 | 10 | 3.0 | 93.69% |
| 2 | 10 | 10 | 3.0 | 90.60% |
| 3 | 30 | 10 | 3.0 | 94.80% |
| 4 | 30 | 20 | 3.0 | 95.61% |
| 5 | 30 | 30 | 3.0 | 95.25% |
| 6 | 30 | 20 | 2.0 | 95.34% |
| 7 | 30 | 200 | 2.5 | 95.16% |

Table 1. Table of accuracies for different ANN configurations

The different values were chosen for trying to maximize the accuracy of the detection of the digits. Considering the given table, the best configuration is using 30 neurons, 20 epochs, and 3.0 as the learning rate; this could be greatly improved but the taken basis was 30 neurons, 30 epochs, and 3.0 as the learning rate (this was used by Nielsen).

**References:**

Bernard, S. 2009. *Using Random Forest for Handwritten Digit Recognition*. HAL. Recovered from: https://hal.archives-ouvertes.fr/hal-00436372

Burges, C. Lecun, Y. Cortes, C. 1998. *The MNIST Database of handwritten digits.* Google Labs, New York. Recovered from: http://yann.lecun.com/exdb/mnist/

Gibiansky, A. 2014. *Speech Recognition with Neural Networks*. Recovered from: http://andrew.gibiansky.com/blog/machine-learning/speech-recognition-neural-networks/

Nielsen, M. 2016. *Neural Networks and Deep Learning*. Recovered from: http://neuralnetworksanddeeplearning.com