Demonstration of Artificial Neural Network’s   
ability to learn and recognize CAPTCHAs

Thanh Nguyen Tien Thien   
20107960

*Abstract*—This report will focus on using Multi-Layer Percep-tion function inside Weka in order to simulate machine’s ability to read and identify noise-added images of 10 different letters drawn in a 10x10 grid and determine the point where it is impos-sible to further recognize the images generated.

Keywords—Artificial Neural Network (ANN), Completely Automated Public Turing Test to tell Computers and Humans apart (CAPTCHAs), Bulletin Board Forums (bbBoard), Random Number Generator (RNG).

# Introduction

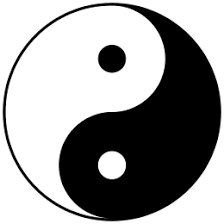


Fig. : Symbol of Yin and Yang

“In Eastern thoughts, the two complimentary forces that make up all aspects and phenomena of life. Yin is the symbol of earth, femaleness, darkness, passivity, and absorption. It is present in even numbers, in valleys and streams, and is repre-sented by the tiger, the colour orange, and a broken line. Yang is conceived is conceived of as heaven, maleness, light, activi-ty, and penetration. It is present in odd numbers, in mountains, and is represented by the dragon, the colour azure, and an un-broken line.”

From (Brittanica)’s exception, the above passage is Figure 1’s definiton of the Yin and Yang symbol, a well known symbol originates from Traditional China that is equivalent to a Western proverb “There are two sides to every coin”. The icon is also a symbol of the beginning of change, of competition, which by the end, pushes the world forward and create better things in life.

In computer science, changes and competitions has long been a popular topic, because the results that it brings to our society has led us to have a better life. Without it, people will stop asking and thinking what can they do to make things easi-er, better, and even cheaper, which ends up killing the strive for innovative ideas to be born and made. A popular example of competition is the battle between Intel and AMD, the two most widely-used Desktop Processor maker. Without AMD rolling out 64-bit microprocessor architecture version of the x86 ins-truction set, we might still be using 32-bit processor from Intel.

The same can be said with humans and technology. The third and fourth industrial revolution saw the rise of electronic devices and the born of IT-related fields. If someone were to tell us 100 years ago, during the early 20th century, in the middle of the se-cond industrial revolution, that one day we will be able to re-place humans on production and automate the whole produc-tion process, almost nobody will believe and think that it is lu-dacris.Yet all the inventions from before and during the 20th century, from the genius inventors, have managed to shape the way we live and communicate with each other. For example, thanks the computer, which is the most important invention of the 20th century, we have been able to digitally communicate with our loved ones, that are geogra-phically disconnected, and even able to see their face through video-calling.

In terms of industrialization, we were able to use computer-controlled hand-like metal grips that we call “robots” to automate manufactu-ring process, and partially or complete-ly replacing humans in the process. Thus, the term “automation” is born to define that process. Automation exists everywhere, especially in any repetitive tasks that a person would want someone to do the job for them. For example, in normal e-veryday computer use, a person might want to repeat a task that requires clicking a bunch of buttons over and over again. Hence “autoclick” program such as (goldensoft.org)’s GS Auto Clic-ker would repeatedly click those buttons mapped on the screen on a user-defined interval.

On the other hand, people can use automation to cause da-mage. (Oxford Advanced Learner's Dictionary) defined da-mage as “physical harm caused to something which makes it less attractive, useful or valuable”. In our case, one commonly-known type of damage is “spamming” (not to be confused with the brand of canned cooked pork). While (Oxford Learner's Dictionaries) might define the word as “the practice of sending mail, especially advertising email, through the internet to a large number of people who have not asked for it”, we shall focus on (Wikipedia, 2021)’s definition of the word, in terms of “un-solicited or undesirable electronic messages”, as simply “sending the same message over and over (again)”. For exam-ple, a person might use automation to spam advertising posts in bullettin board forums to seek attention for contents inside the post.

In order prevent bots message-spamming, Completely Au-tomated Public Turing test to tell Computers and Humans A-part (CAPTCHA) has been introduced. Basically, it is a test that a human must give the correct input inside an image that contains a sequence of random numbers and integers that have noises added. Because those noises changed the pixels that made up some words, it is harder to get the words inside a CAPTCHA-generated image in a simple image recognition system. This has proven to be an effective way to prevent bot-spamming and it is still being commonly used across systems and forums.

As a person who keeps wanting to be an attention whore for the purpose of gaining money for advertising action, I must be able to come up with a way that would be able to recognize the keywords inside CAPTCHAs. Luckily, we were able to partial-ly beat the test, by using Artificial Neural Network (ANN) to train and recognize the jumbled letters. I will use Multi-layer Perception function from Weka to help me with the demon-stration for this paper.



Fig. : A generated CAPTCHA (from fakecaptcha.com)

# Artificial Neural Network Design, architecture & parameters

This section will go through a brief explanation of Artificial Neural Network, as these will be what this paper will use for the demonstration purposes.

As I am not a data scientist myself, and there are people out there that can explain about what these are and how these work better, shorter and easier than myself can, thus the explanation will be based on the cited work, along with my further com-ments to show my understanding about the topic.

## Artificial Neural Network

### Design

This part will touch on the basics of what ANN is. (Simplilearn, 2019) has a quick 5-minute sum-up video about this topic that I shall cite for the quick explanation of this topic:

“Neural Network form the base of deep learning, (which is) a subfield of machine learning where the algorithms are inspi-red by the human brain. Neural networks take in data, train themselves to recognize the pattern in this data and then predict the output for a new set of similar data”.

In other words, Neural Network is a tool that simulates the human brain in order to do certain repetitive tasks that usually requires human intelligence. This tool can be viewed like a child trying to learn what the education system wants them to, by providing them with a list of similar-patterned data, and tell them that what do they need to learn from the data and what would be expected from it.

In real-world applications, Neural Network can be found to be the core of certain tasks in the form of technology. For ex-ample, Facial Recognition and Live Text-translation.

### Architecture

An ANN has 3 layers: input, hidden and output layer. The input layer, can be identified by the left-most column in Weka GUI and coloured in green squares, specifies the variables that shall help identify the characteristics that would help the machine recognize. For example, if I want to train my machine to recognize different types of Iris-coloured flowers, then I would want variables such as the width or the length of the flower’s sepal and petal. In this case, each pixel in the grid is defined in the range of 0 and 1, equivalent to “on” and “off”, or there is texture inside each pixel, together forming a total of 100 pixels. The red circle in the middle of the GUI is called the hidden layer, and how many of them specifies the number of available computation nodes for the function. Most of the cal-culation also takes place in here Finally, the output layer, usu-ally named “class”, can be identified by the yellow-coloured circle, defines a list of possible output for the datasets.

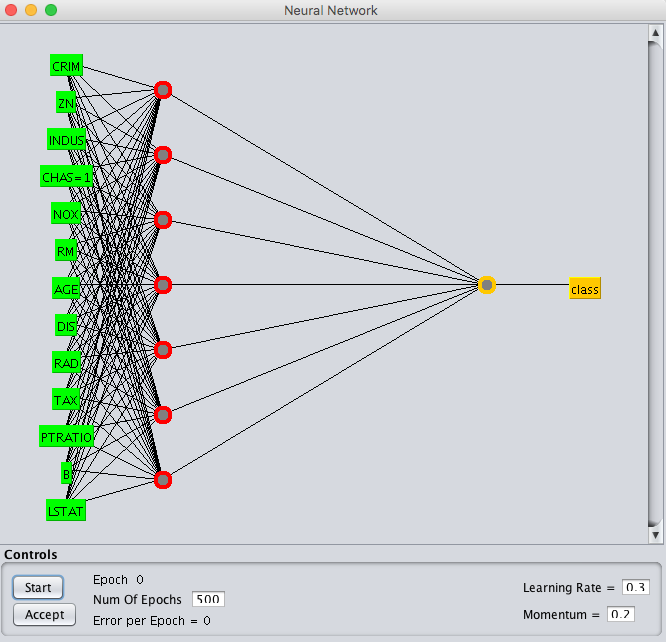


Fig. : Weka GUI Designer for the Multi-Layer Perception Algorithm (Brownlee, 2016)

However, as previously stated, in order to identify the correct words, most of the calculation takes place in the hidden layer. (Simplilearn, 2019) also gave a pretty good, quick, de-tailed and easy-to-understand explanation that I shall quote from:

“Each pixel is fed as input to each neuron of the first layer. Neurons of one layer are connected to neurons of the next layer through ‘channels’. Each of these channels is assigned to a nu-merical value, known as ‘weight’. The inputs are multiplied to the corresponding weights and their sum is sent as input to the neurons in the hidden layer. Each of these neurons is associated with a numerical value called the ‘bias’, which is then added to the input sum. This value is then passed through a threshold function called the ‘activation function’. The result of the acti-vation function determines if the particular neuron will get activated or not. An activated neuron transmits data to the neurons of the next layer over the channels. In this manner, the data is propagated through the network. This is called ‘forward propagation’. In the output layer, the neuron with the highest value fires and determines the output.”

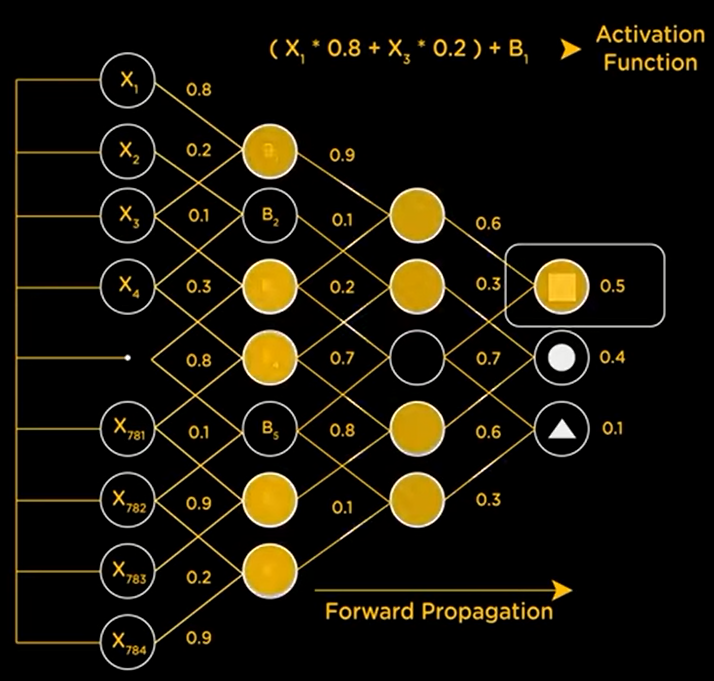


Fig. : Diagram of Forward Propagation process to determine the correct class (Simplilearn, 2019)

The values that are used to identify the correct set of class are basically a probability. Each of the values are cross-validated with the neurons of the next layer through the channels in order to determines which neurons to activate, and then finally the neurons with the largest probability is used to determine the output value.

However, without training, the network won’t be able to co-rrectly identify the given data. Hence the process of Back Pro-pagation, which is the process of balancing the probability values. Basically, the list of classes are then balanced by the actual output error by increasing or decreasing the probability values.

### Parameters

When processing the data with ANN, there are 3 different types of parameters: ARFF Attributes’ possible data types, Weka’s processing, and the parameters of answers.

#### Parameters to process the data

This section refers to a brief explanation of how Weka’s .arff (Attribute-Relation File Format) works, since this file is the basics of how Weka will read and train the Network, it also expects what the parameters should be. (waikato, 2008) hosts the wiki page for this file format.

For basic structures of file: @RELATION defines the name of the data sets, while @ATTRIBUTE defines the variables, or characteristics, to identify the objects, and the list of objects that the user wants the network to identify. Finally, @DATA should be self-explanatory.

For @ATTRIBUTE, one field can accept 4 different data types: numeric, string, date, and nominal specification.

1. Numeric: This can be a real or integer number. E.g. @ATTRIBUTE p00 NUMERIC.
2. String: Useful in text-mining applications, this allows us to create values that contains arbitrary textual values, defined by the following format: @ATTRIBUTE username string.
3. Date: Use to process time, by default use ISO-8601 but custom format can be specified. E.g. @ATTRIBUTE timestamp DATE “yyyy-MM-dd HH:mm:ss”.
4. Nominal attributes: Similar to an array in programming, a possible list of hard-coded values can be specified, e.g. @ATTRIBUTE gender {Male,Female}, or could even be @ATTRIBUTE pixelData {0,1}.

The last section is @DATA, @DATA defines the values that form the object given by the list of attributes. For example, the above Date attribute would look something like this: “2021-12-03 19:15:21”.

#### Parameters to determine the correctness of the network towards the sets.

In order to determine the rate of which the network is able to correctly identify, terms such as TP (True Positive) Rate and FP (False Positives) Rate are used, and both of which can be understand by the visualization of this chart:

|  |  |  |
| --- | --- | --- |
|  | Correct answers | |
| Machine Predicts | True Positives | True Negatives |
| False Positives | False Negatives |

Table : Sorting types of Network’s Results.

Therefore, TP Rate is used to determine the percentage of which machine predicts correctly the number of True Positives, where FP rate is a direct opposite to TP Rate, where it counts the rate of False Positives.

Next are Precision and Recall. (Wikipedia, 2021) gave a good explanation about these 2 values, but basically to sum up, “Precision can be seen as a measure of quality, and recall as a measure of quantity.”

The formula of precision is calculated by the fraction of retrieved documents that are relevant to the query:

(Wikipedia, 2021)

While recall is calculated as the fraction of the relevant documents that are successfully retrieved:

(Wikipedia, 2021)

Another 2 terms that are used to specify the correctness of the Network are “Mean Absolute Error” and “ROC Area”. “ROC Area” is, in layman terms, a graphing of Precision and Recall, while (C3 AI) explained “Mean Absolute Error” as “the magni-tude of difference between the prediction of an observation and the true value of that observation”, that is, comparing to the true value, how much difference is the prediction it is compared to the original value.

#### Parameters to determine Weka’s ARFF attributes.

In Weka, “NominalToBinaryFilter, NormalizeAttributes, and NormalizeNumericClass” is used to turn numerical values (values where data sets are larger than 1 or smaller than 0), into values of 0 and 1. However, in my test, since most of my values are 0 and 1, these two options are not needed at all.

LearningRate and Momentum is used to control the Net-work’s speed of learning set in a given TrainingTime. Training time is number of generations that the machine will take time to learn.

HiddenLayers is used to control the number of Neurons that are available to the machine. This directly affects the learning process.

# Generating training and test sets

Before jumping into how the sets are generated, an expla-nation is needed on the format of the sets. As previously men-tioned, Weka will be used for the demonstration, therefore the input file must be designed for this program to work. As this uses “.arff” (Attribute-Relation File Format), I personally found it very similar to the structure of Pascal, hence a brief reading on the .arff file format Wiki easily helps me create the structure for the files.

Since I will be using a 10x10 grid to draw the words, I shall define 100 pixels and each picture shall holds 2 values, 0 and 1. Therefore my attribute should be follow the structure of “@ATTRIBUTE <name> {possible, value}”, where <name> would stands for the name of characteristics that the network needs to identify, and in this case, they follow the format of “p09”, where “p” stands for the word “pixel”, “0” stands for the rows, “9” stands for the columns. Thus, “p90” should stands for “pixel at row 9, column 0” (bottom left corner). {possible, value} holds the list of possible values for that characteristics, for example in this case each pixel only has 0 or 1, thus it should be “0,1”. Therefore, an attribute should be like “@ATTRIBUTE p90 {0,1}”, and each attribute is defined in a separate new line. Finally, For @DATA, a letter A in my data sets shall have the possible values defined in each pixels from 00 -> 99 (and follows by columns first then to rows):

0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,1,1,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,A

Therefore, there are total of 101 attributes, including 100 attributes to define the value for each pixel, and the last “class” value defines which word it is. There are only 10 classes, which are 10 letters ranging from A to J, will be tested in this case.

## Training sets

For each word, there are total of 8 different fonts created. This ranges from different shapes, widths and fonts of the letter. Plus, to help with the visualization of the letter, a small C++ program is used, where it would draw a 10x10 (well actually grid width and length is defined in the source code itself and can be changed, but the program must be re-compiled) grid, and selecting the pixel location in formats of ij (i stands for row while j stands for column, e.g. 09 would select top-left pixel) would flip the pixel of that location back, and when finished, the program would output the data for me to put into Weka.

A screen shot of a computer

Description automatically generated with low confidenceCalendar

Description automatically generated with low confidence

Fig. : Different styles of letter “A”

## Test sets

There are total of 4 tests for the demonstration: 10%, 20%, 30% mutated, and Periodic Test. In order to generate test sets, 1 font out of 8 is chosen from each letter of the training sets to mutate for each tests. However, to prevent bias, Random Num-ber Generator (RNG) from the website Random.org is used to select the words. Plus, when selecting which letter to use for the test, I must not use the font that has been used for the previous test.

The percentage mutation test is similar to a picture’s salt-and-pepper noise, where random bits of 0 is flipped into 1 and vice-versa. Again, to prevent bias, RNG is once again used to determine the pixel location that I need to flip, ranging from 00 to 99. If the pixel location contains a 0, I must flip it into 1 and vice-versa. 10-20-30% stands for the percentage of data must be mutated. In this case, for example, 10% mutated means 10 out of 100 pixels must be flipped, while 20 means 20 out of 100 pixels and so on.

Calendar

Description automatically generated with low confidence

Fig. : Letter “B” with 10% of total pixels flipped

Periodic test, however, is inspired by the periodic noise in an image, where lines would appear in an image as a result of a electrical or electromechanical interference during image capturing process. In this case, the whole line of that character would be turned into either 0s or 1s, and this would happen alternatingly, which means the whole rows of 1-3-5-7-9, or 0-2-4-6-8 would be flipped, not just one specific line. This test might sound out of place but this is a demonstration of a test where the noises are patterned, while mutation tests are noises that doesn’t happen frequently. Again, in order to also prevent bias, RNG is used to generate a list of 0s and 1s, which indicates odd or even rows should be flipped, and all the data should be flipped entirely into 0 or into 1.

A group of horses in front of a large building

Description automatically generated with low confidence

Fig. : An image with Periodic Noise

A screen shot of a computer

Description automatically generated with low confidence

Fig. : Letter “D” with Periodic Noise added

Finally, in order to view and cross-validate with the data that I generated, I have put up the Excel sheet for the data that I generated into GitHub, with the same “Assignment 2” repo that I uploaded. This can be found by going to “DataLog” folder of the repos and open the corresponding Tests file.

# Experimentation and Results

Before I begin, Weka logs for the test cases can also be found on the GitHub that is linked above, on the “ResultsLog” folder, if in case any verification purpose is needed.

The procedure to train is as follows: First, the network will be fed with TrainingSet.arff that contains the total of 80 models from A to J. Then, the Network will be tested with images that are 10, 20, 30% mutated respectively, and finally the periodic test images. Not once in any case were the tests gets interrupted (as if all the tests were done in one run, and continuously), nor the sets were re-trained with the test sets.

### 10% mutation & 20% mutation

For both 10% and 20% mutation, the Network can recog-nize all 10 letters, with 100% accuracy. Therefore, these two test cases are grouped together. However, the only notice that were recorded were “mean absolute error” being increased between the two tests. This means that the machine takes significantly longer in order to completely recognize the characters correctly. Specifically, 10% and 20% have a Mean Absolute Error of 0.009 and 0.0399, respectively.

### 30% mutation

The Network showed cases of unable to recognize certain characters in this case. With 30%, the recorded TP rate and FP rate is 0.6 and 0.044, respectively, and Recall of 0.6 and ROC Area of 0.933. The machine cannot calculate Precision value, be-cause the lack of “False Positives” in this case, only “Right” or “Wrong”.

The following is the confusion matrix for the 30% mutation:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | 🡨 classified as |
| 1 |  |  |  |  |  |  |  |  |  | A = A |
|  | 1 |  |  |  |  |  |  |  |  | B = B |
|  |  |  |  |  |  |  |  |  | 1 | C = C |
|  |  |  | 1 |  |  |  |  |  |  | D = D |
|  |  |  |  |  |  |  | 1 |  |  | E = E |
|  |  |  |  |  | 1 |  |  |  |  | F = F |
|  |  |  |  |  |  | 1 |  |  |  | G = G |
| 1 |  |  |  |  |  |  |  |  |  | H = H |
|  |  |  |  |  |  |  |  |  | 1 | I = I |
|  |  |  |  |  |  |  |  |  | 1 | J = J |

Table : Confusion Matrix of 30% mutation

This table showed that the letter C is being recognized as J, E is recognized as H, H is recognized as A, and I is recognized as J. That gives it 4 answers being predicted incorrectly.

### Periodic Test.

The purpose of adding periodic test is because unlike muta-tion test where the bits being flipped are random, the noise added into the image are constant, i.e the location of noise is always at that one specific location with the same pattern. Hence, the ma-chine can be trained in order to neglect these noises and construct a smaller word from view. However, for the purpose of testing untrained network for these unexpected cases, this noise will not be trained at all but be read by the machine immediately.

For this test, the TP and FP rate is reduced to 0.5 and 0.056, recall of 0.5 and ROC Area of 0.944.

The following confusion matrix showed that the letter A and F is recognized as letter H, while D, E, and G is recognized as B. With 5 out of 10 cases being recognized correctly, this makes the test only has 50% accuracy.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | G | G | H | I | J | 🡨 Classified as |
|  |  |  |  |  |  |  | 1 |  |  | A = A |
|  | 1 |  |  |  |  |  |  |  |  | B = B |
|  |  | 1 |  |  |  |  |  |  |  | C = C |
|  | 1 |  |  |  |  |  |  |  |  | D = D |
|  | 1 |  |  |  |  |  |  |  |  | E = E |
|  |  |  |  |  |  |  | 1 |  |  | F = F |
|  | 1 |  |  |  |  |  |  |  |  | G = G |
|  |  |  |  |  |  |  | 1 |  |  | H = H |
|  |  |  |  |  |  |  |  | 1 |  | I = I |
|  |  |  |  |  |  |  |  |  | 1 | J = J |

Table : Confusion Matrix for Periodic Noise Test

# Conclusions

To conclude, even though there is a limitation to the percen-tage of the mutation that the machine can read, we can also say that this limitation also exists as human being. This means that, if as humans, given a certain percentage of a mutated picture, that we cannot even be able to read the picture, then there is a possibility that the machine cannot also be able to read that same picture. Since the test fails off at 30%, then there is a possibility that at around 25% (which is a quarter of the image is mutated), neither the bot or human can recognize the pictures correctly.

These test cases, however, is tested based on a given pixel got damaged by the noise being added. In real-world scenario, there might be a possibility where the word is dented, as given in Fig. 2, or noise being added on top of a given pixel. Therefore, it might be possible to train the bots in order to identify and neg-lect these type of noises. Overall, if it is possible to at least neg-lect some noises from images, then it is not impossible to also train the Network to do these things, as previously mentioned with Periodic Noises.

The test cases can be more interesting if words that got dented can be tested, but due to time constrains, this test case has to be neglected.

Finally, the source code and the GitHub project can be found at [baddles/Assignment-2 (github.com)](https://github.com/baddles/Assignment-2).

# References

(n.d.). Retrieved from https://www.oxfordlearnersdictionaries.com/definition/english/damage\_1

baddles. (2021, December 3). *Assignment 2 files.* Retrieved from GitHub: https://github.com/baddles/Assignment-2

Brittanica. (n.d.). *yinyang Introduction & Quick Facts.* Retrieved from Brittanica: https://www.britannica.com/topic/yinyang

Brownlee, J. (2016, July 22). *How To Use Regression Machine Learning Algorithms in Weka.* Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/use-regression-machine-learning-algorithms-weka/?ref=upflow.co

C3 AI. (n.d.). *Mean Absolute Error.* Retrieved from C3.ai: https://c3.ai/glossary/data-science/mean-absolute-error/

goldensoft.org. (n.d.). *goldensoft.org*. Retrieved from GS Auto Clicker: https://goldensoft.org/

Goldensoft.org. (n.d.). *goldensoft.org*. Retrieved from GS Auto Clicker: https://goldensoft.org/

Oxford Advanced Learner's Dictionary. (n.d.). *Definition of damage noun from the Oxford Advanced Learner's Dictionary.* Retrieved from Oxford Learner's Dictionaries: https://www.oxfordlearnersdictionaries.com/definition/english/damage\_1

Oxford Learner's Dictionaries. (n.d.). *Definition of spamming noun from the Oxford Advanced Learner's Dictionary.* Retrieved from Oxford Learner's Dictionaries: https://www.oxfordlearnersdictionaries.com/definition/english/spamming

Simplilearn. (2019, June 19). *Neural Network In 5 Minutes | What Is A Neural Network? | How Neural Networks Work | Simplilearn.* Retrieved from YouTube: https://www.youtube.com/watch?v=bfmFfD2RIcg

waikato. (2008, November 1). *Attribute-Relation File Format (ARFF).* Retrieved from waikato: https://www.cs.waikato.ac.nz/ml/weka/arff.html

Wikipedia. (2021, October 18). *Precision and Recall.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Precision\_and\_recall

Wikipedia. (2021, November 23). *Spamming.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Spamming