# ANNs for diagnosing Breast Cancer

Stavros El. Alexiou, Student
Computer Engineering Department (CEID)
University of Patras
Greece
st1059680@ceid.upatras.gr

Dr., M.Sc., George - Peter K. Economou
Computer Engineering Department (CEID)
University of Patras
Greece
gpoikonomou@ceid.upatras.gr

***Abstract*—By this work, we utilized established medical records and via several Artificial Neural Network Architectures and varied training algorithms we show the viability of each model and the potential of such records. The paper treats our efforts to map the medical data, obtained good results, and offer an insight in Artificial Intelligence applications.**

***Keywords—SOM, feed forward / recurrent Artificial Neural Networks, medical databases, data mapping, breast cancer***

## I. Introduction

Artificial Intelligence refers to the ability of a machine being able to reproduce a person's cognitive functionality, such as learning, designing, and decision making. Human brains processing information differently than a conventional digital computer was a key motivator for more research in Artificial Neural Networks (ANNs). In particular, by using appropriate models and recognizing various patterns it is possible to have viable solutions of complex problems.

ANNs' models are composed of many non-linear identical units operating in parallel and arranged in patterns similar to biological neural networks. Their key element, extensive parallelism, is essential for high performance speech / image recognition, decision support, as well as the diagnosis of illnesses requiring Medical Doctors' (MDs) excessive training.

Predicting breast cancer diagnosis, given a dataset extracted from a computed digitized image of a fine needle aspirate (FNA) of a breast mass, by means of ANNs architectures, is the topic treated in this paper. By looking at the image and the features extracted, a decision has to be made, either this breast mass being *benign (B)* or *malignant (M)*, essentially making the matter to be dealt with a two-class problem.

Deep learning artificial neural networks were chosen as the appropriate experiment models. They are trained via commonly established percentages (90% to 60%) of the provided dataset features. The rest of the dataset (10% to 40%) is used as accuracy metrics in order for the ANNs to diagnose patients' symptoms from data they were subjected to.

Data provided first had to undertake pre-processing in order to be mapped in patterns' set of artificial neurons inputs / outputs (I/O). Also, Self-Organizing Maps, a special ANNs' class were employed to detect possible data intra-relations.

## II. Data (pre-)Processing and Model Construction

In order to fit to the ANNs' model (architecture and training algorithm), understanding how the data are mapped as I/O patterns is a key factor. Basic python libraries were put to use, such as <pandas>, <numpy>, <seaborn>, <matplotlib>, <sklearn> (for pre-processing) and <model_selection> for visualization and data handling plus <keras> models and layers for model construction, compiling and finally fitting [1].

Medical data were obtained from an established database, consisting of 569 different entries each with 32 features [2]. Hence, their intra-relations were sought out by Kohonen's self-organizing maps (SOM) [3]. SOM's, neural net-based dimensionality reduction algorithms is used to represent a given dataset as a two-dimensional discretized pattern. Fig. 1 shows a part of the ensued classification.

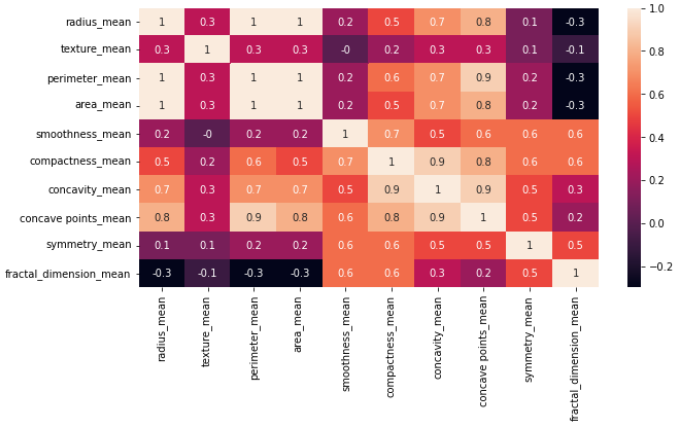


Fig. 1. Self-organizing map of medical data major areas

As the reader can see, similarity percentages were very low; thus every conducted experiment since, included random samples selection in equal percentages from all data.

Moreover, handling I/O patterns' missing values is also crucial since when dealing with structured data, missing values are an inevitable occurrence [4]. Such values can reduce the statistical power of a study producing biased estimates and leading to invalid conclusions. Perhaps the field was not applicable and the data was not available or even the person who entered the data did not know the correct value.

Two commonly applied methods comprise either dropping these values or staging suitable ones ("*s_value*"). A general used formula, "*s_values*", is described below [5]:

$$s\_value = 0.50 * [length(value's\ feature\ scale)] \quad (1)$$

After the medical database was loaded on the python environment as a "dataframe", via the <pandas> library, no such values were found at all by tracing missing values.

Mapping data, floating point numbers were given to every feature type with the exception of the output; it was handled as binary, because it could only be either "*B*" or "*M*", easily utilizing <Label Encoder> from <sklearn>'s pre-processing library. Also, there was no need for balancing the I/O patterns according to the two outputs' present percentages.

Data representation was subsequently dealt. Values were in <tens>, <hundreds>, and even in <one-tenth> or <one-hundredth>; some wide scale values. Thus, they were normalized by utilizing <sklearn>'s "MixMaxScaler" feature. As the name suggests, scaling receives a minimum and a maximum value from a certain range of numbers and transforms them in a ratio between *zero (0)* and *one (1)*. Fig. 2 shows a selection of applied scaled values.

```
[[0.52103744 0.0226581  0.54598853 ... 0.91202749 0.59846245 0.41886396]
 [0.64314449 0.27257355 0.61578329 ... 0.63917526 0.23358959 0.22287813]
 [0.60149557 0.3902604  0.59574321 ... 0.83505155 0.40370589 0.21343303]
 ...
 [0.45525108 0.62123774 0.44578813 ... 0.48728522 0.12872068 0.1519087 ]
 [0.64456434 0.66351031 0.66553797 ... 0.91065292 0.49714173 0.45231536]
 [0.03686876 0.50152181 0.02853984 ... 0.         0.25744136 0.10068215]]
```

Fig. 2. Some apllied scale values

The final step was splitting the given dataset in two, a training and a testing dataset populated as stated above. With the first set the model was to be trained with, while the latter, constituting a set of features ANNs were never exposed to, was used for measuring each tested model's accuracy.

The first exploited ANNs' model was the *Sequential* type from <keras>' models library. The input consisted of 30 artificial neurons (AN), known as "*Input Layer*"; each of their outputs fed the consecutive one, called "*Hidden Layer*". Every AN, except input ones, were composed of some weighted inputs, a transformation function and an activation function, corresponding to a biological neuron's axon. Between the "*Input Layer*" and the "*Hidden Layer*" a random 20% drop of the initial connections was set, in order to avoid overfitting issues. This occurs when a statistical model fits exactly with regards to its training data, thus resulting in the algorithm not being able to perform accurately against previously not fed data [6]. A number of empirical methods were then tried so as to define the number of hidden layers and the number of ANs they. Only 1 hidden layer, consisting of 16 ANs was chosen. The last layer, predictably, was set to have 1 AN.

A recurrent Neural Network (RNN) [7] model was also tried. RNNs are composed of many internal hidden states that are updated every time a new input is received. Those states are then fed back to the RNN by the time it reads a new input.

In addition, RNNs produce outputs at every time step, i.e. it reads an input, updates its hidden state, this is fed again to it and then it produces a new output. The functional form of this recurrence relation is described from the following equation.

$$h_t = f_w(h_{t-1}, x_t) \quad (3)$$

$<h_t>$ is recurrently computed at each time step by means of a function $<f_w>$ that depends on some weights $<w>$, while accepting as input the previous hidden state, $<h_{t-1}>$ and the current state $<x_t>$. Time steps are arbitrarily chosen.

Inputting the medical data to the RNN proved to be a major problem, leading to poor testing results (roughly a 60% average correct diagnoses when fed training sets), due to the given dataset's dimensionality. The solution was to build a model by exploiting the "*many-to-one*" procedure [8]. Scaling process done, as aforementioned, a 3-D array was produced from the initial 2-D one by utilizing <numpy>'s "*reshape*" method. Thus the 3-D array consisted of 569 2-D sub-arrays of 30 records per feature. Each of the 2-D sub-arrays (ranked 30 x 1) is related to its corresponding output value ("*B*" or "*M*").

Implementing the RNN, <keras>' library was put to use exploiting the Long Short-Term Memory (LSTM) for a number of its units. LSTM is a variation of RNN networks where a unit's architecture includes a "memory cell" that can maintain data information for longer periods of time than standard RNN variations. LSTM was used in the initial layers of the model while its last one was a fully connected or "*dense*" layer. Its first layer consisted of 569 ANs, each one of them receiving 30 data inputs and resulting to 1 binary output. After again a 20% random connection drop, these 569 outputs led to the next LSTM layer which consisted of 128 ANs carrying the received sequences. Another 20% random connections drop was performed thus leading to the last layer with just 1 AN.

## III. Results

In this section results regarding our experiments are given. Fig. 3, 4 show random samples of training and validation accuracy curves for the feed forward ANN. The medical dataset was split by a 60% - 40% ratio of training vs. testing patterns. The model was arbitrary set to run for 100 epochs.


Fig. 3. Sampled training and validation curve for the feed forward ANN


Fig. 4. Sampled training and validation curve for the feed forward ANN

Random samples concerning the same curves for the RNN are shown in Fig. 5, 6. Notice the very dissimilar training / testing highs and lows of the curves, between the two models, plus the much better final conversion of the RNNs.


Fig. 5. Sampled training and validation curve for the RNN


Fig. 6. Sampled training and validation curve for the RNN

All and all much can be said about the consistency of the medical dataset, the precision of randomly splitting it into two sets, training and testing (60% - 40% ratio), and the good end results (well over 95% validation accuracy).

It also should be said that by using a PC equipped with MS-Windows v10Pro, Intel Core i3@3.70GHz, and 16GB of RAM, 100 epochs averaged to 10.67seconds for the feed forward ANN and 271.78seconds for the RNN when the networks where trained (again, with the 60% of the dataset).

Table 1 shows average times for running the feed forward ANN and its training and testing (60% - 40% ratio) epochs.

| 60/40 | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th | 10th | AVG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| loss | 0.2526 | 0.3203 | 0.2538 | 0.2643 | 0.2888 | 0.3146 | 0.2337 | 0.2384 | 0.2700 | 0.3578 | **0.2797** |
| accuracy | 0.9265 | 0.8824 | 0.9069 | 0.9216 | 0.8971 | 0.9020 | 0.8971 | 0.9265 | 0.9265 | 0.8627 | **0.90493** |
| validation loss | 0.2566 | 0.2815 | 0.2203 | 0.2840 | 0.2786 | 0.2523 | 0.2443 | 0.2026 | 0.2668 | 0.3201 | **0.26071** |
| validation accuracy | 0.9416 | 0.9197 | 0.9562 | 0.9051 | 0.9270 | 0.9416 | 0.8978 | 0.9416 | 0.9343 | 0.8759 | **0.92408** |
| **Average Time = 10.67 sec** | | | | | | | | | | | |

Table 1: Average training / testing times for the feed forward ANN

Table 2 shows average times for running the RNN and its training and testing (60% - 40% ratio) epochs.

| 60/40 | 1st | 2nd | 3rd | 4th | 5th | 6th | 7th | 8th | 9th | 10th | AVG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| loss | 0.2406 | 0.2593 | 0.2196 | 0.2333 | 0.2881 | 0.2400 | 0.1991 | 0.2679 | 0.2166 | 0,1489 | **0.23134** |
| accuracy | 0.9118 | 0.8873 | 0.9314 | 0.8971 | 0.8873 | 0.9069 | 0.9118 | 0.8922 | 0.9167 | 0.9363 | **0.90788** |
| validation loss | 0.2602 | 0.2046 | 0.2551 | 0.3065 | 0.2382 | 0.1769 | 0.2210 | 0.2041 | 0.1997 | 0.3331 | **0.23994** |
| validation accuracy | 0.9197 | 0.9416 | 0.9051 | 0.8613 | 0.8978 | 0.9270 | 0.9270 | 0.9270 | 0.9416 | 0.8905 | **0.91386** |
| **Average Time = 271.78 sec** | | | | | | | | | | | |

Table 2: Average training / testing times for the RNN

## IV. DISCUSSION

The aim of our research was to test the effectiveness of crucial medical data found in an internet site, test them by a number of Artificial Neural Network architectures and training algorithms, reach feasible results and compare them.

It can be said that although we present a work in progress, initial results are far more than simply satisfying. In comparison with previous works [5], PCs have immensely more computational power, programming environments offer a number of ready-to-use free libraries that can decrease programming time with reliable calculation products, and a number of more effective ANN architectures and training algorithms have appeared and continue to emerge [9, 10].

All things considered, the time that a full human body diagnose model, based on Artificial Intelligence Techniques, such as ANNs, Fuzzy Logic, etc., is soon to be developed, too. As the saying goes, “there are not illnesses, there are patients”.

## V. CONCLUSIONS

A dataset, consisting of medical features regarding breast cancer diagnoses was acquired from a well-known internet site. Consequently, it was mapped, input, trained, and, finally, tested by a feed forward ANN and a RNN. Results were presented, compared, and analyzed, and a furtherance of our work outlined. Those were first-rated and will be further promoted.

## REFERENCES


[1] https://keras.io/api/

[2] https://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/cancer/WDBC/

[3] T. Kohonen, "The self-organizing map" in Proceedings of the IEEE, vol. 78, no. 9, pp. 1464-1480, Sep. 1990.

[4] Hyun K., "The prevention and handling of the missing data" in Korean J Anesthesiol., pub. online, May 2013.

[5] Economou, G. - P. K., Decision Support Systems for Tele-Medicine Applications, Research Studies Press Ltd., Hertfordshire, UK, 2004.

[6] Xue Y., "An Overview of Overfitting and its Solutions", in Journal of Physics Conference Series, vol. 1168, pp. 1-6, Feb. 2019.

[7] https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[8] https://wandb.ai/ayush-thakur/dl-question-bank/reports/LSTM-RNN-in-Keras-Examples-of-One-to-Many-Many-to-One-Many-to-Many---VmlldzoyMDIzOTM

[9] Bharati S., Podder, P., and Hossain Mondal, M. R., "Artificial Neural Network Based Breast Cancer Screening: A Comprehensive Review", Int. J. of Comp. Inf. Syst. and Ind. Man. App., vol. 12, pp. 125-137, 2020.

[10] Anji Reddy, V. and Badal Soni, B., "Breast Cancer Identification and Diagnosis Techniques", Mach. Learn. for Int. Dec. Science, pp. 49–70, 2020.