A Novel Chaos Theory Inspired Neuronal Architecture

Harikrishnan N B
Consciousness Studies Programme
National Institute of Advanced Studies
Indian Institute of Science Campus, Bengaluru, India
harikrishnan.nb@nias.res.in

Nithin Nagaraj
Consciousness Studies Programme
National Institute of Advanced Studies
Indian Institute of Science Campus, Bengaluru, India
nithin@nias.res.in

***Abstract*—The practical success of widely used machine learning (ML) and deep learning (DL) algorithms in Artificial Intelligence (AI) community owes to availability of large datasets for training and huge computational resources. Despite the enormous practical success of AI, these algorithms are only loosely inspired from the biological brain and do not mimic any of the fundamental properties of neurons in the brain, one such property being the chaotic firing of biological neurons. This motivates us to develop a novel neuronal architecture where the individual neurons are intrinsically chaotic in nature. By making use of the *topological transitivity* property of chaos, our neuronal network is able to perform classification tasks with very less number of training samples. For the MNIST dataset, with as low as 0.1% of the total training data, our method outperforms ML and matches DL in classification accuracy for up to 7 training samples/class. For the Iris dataset, our accuracy is comparable with ML algorithms, and even with just two training samples/class, we report an accuracy as high as 95.8%. This work highlights the effectiveness of chaos and its properties for learning and paves the way for chaos-inspired neuronal architectures by closely mimicking the chaotic nature of neurons in the brain.**

***Index Terms*—Chaos, Topological Transitivity, Generalized Luröth Series, Neural Network, Machine Learning**

I. Introduction

Next to the universe, the human brain is the most complex and sparsely understood system. Brain science is said to be in *Faraday stage* [1], which means our current understanding of the working of the brain is at a very primitive stage. It has been estimated that human brain has approximately 86 billion neurons [2] which interact with each other forming a very complex system. Neurons are inherently nonlinear and found to exhibit chaos [3], [4]. An interesting counter-intuitive property of networks of neurons in the brain is their ability to learn in the presence of enormous amount of noise and neural interference [5]. Inspired by the biological brain, researchers have developed Artificial Intelligent systems which use learning algorithms such as Deep Learning (DL) and Machine Learning (ML) that loosely mimic the human brain.

DL and ML algorithms have a wide variety of practical applications in computer vision, natural language processing, speech processing [6], cyber-security [7], medical diagnosis [8] etc. However, these algorithms do not use the essential properties of human brain. One such property of brain is the presence of chaotic neurons [3], [4]. Even though Artificial Neural Networks are biologically inspired, none of its varied architectures have neurons which exhibit chaos though it has been shown that certain type of neural networks exhibit chaotic dynamics (for e.g., in Recurrent Neural Networks [9]). Chaotic regimes with a wide range of behaviors are beneficial for the brain to quickly adapt to changing conditions [3]. There has also been some evidence that weak chaos is good for learning [10]. Inspired by these studies, in this work, we explore whether chaos can be useful in learning algorithms.

There have been previous attempts at developing novel biologically inspired learning architectures. A recent research study by [11] proposes a learning architecture that uses a mathematical model of the olfactory network of moth to train to read MNIST[1] [12]. The same study [11] also highlights learning from limited data samples. In another interesting research [13], the authors propose a novel compression based neural architecture for memory encoding and decoding that uses a 1D chaotic map known as Generalized Luröth Series (GLS) [14]. GLS coding, a generalized form of Arithmetic coding [15], is used for memory encoding in their work.

In this work, we propose for the first time - a novel neuronal architecture of GLS neurons and train it for a classification task using a unique property of chaos known as *Topological Transitivity* (TT). This research is a first step towards building a more realistic brain-inspired learning architecture. Here, chaos is used at the level of individual neurons. As we shall demonstrate, one of the key benefits of our proposed method is its superior performance in low training samples regime.

The paper is organized as follows. We introduce the novel architecture in Section II along with a detailed description of the topological transitivity based classification algorithm. This is followed by experiments and results in section III. We conclude by highlighting the unique advantages of our method while also mentioning some of the possible future research directions in section IV.

II. The Proposed Architecture

The basic diagram of the proposed neural architecture is provided in Figure 1. It comprises of an input layer of GLS neurons represented as $C_1, C_2, ..., C_{mn}$. The GLS neuron is

[1] http://yann.lecun.com/exdb/mnist/index.html

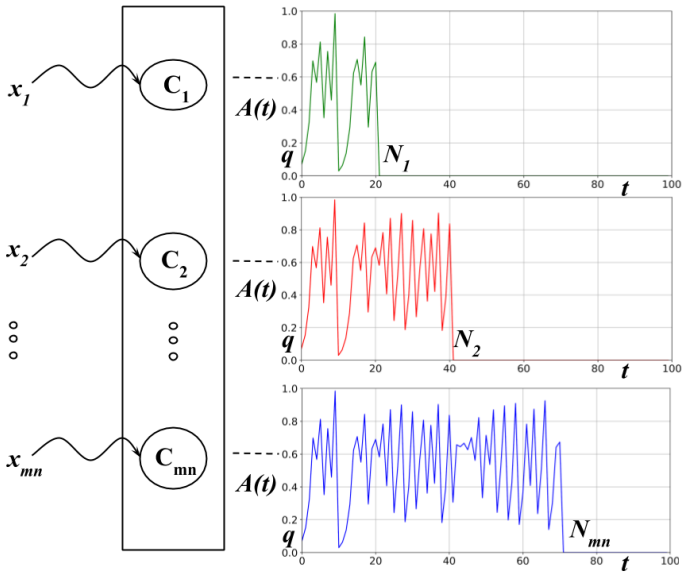

Fig. 1. The proposed Chaotic GLS neural network architecture. $C_1, C_2, ..., C_{mn}$ represent the input layer of chaotic GLS neurons each with an initial normalized membrane potential of q units. $\{x_i\}_1^{mn}$ is the normalized set of stimuli that is input to the network. Each GLS neuron fires (chaotically) until its membrane potential $A(t)$ is in the neighbourhood of the stimulus. The *firing time* N_i ms of the corresponding GLS neuron C_i is the *topological transitivity* based extracted feature.

a 1D chaotic map which we shall describe shortly. The GLS neurons get activated in the presence of a stimulus (input data) which results in a chaotic firing pattern. Each GLS neuron in the input layer continues to fire chaotically until its amplitude matches that of the stimulus - at which point it stops firing. In the model provided in Figure 1, $x_1, x_2, .., x_{mn}$ represents the stimulus (normalized) which is assumed to be a real number between 0 and 1. Each GLS neuron has an initial normalized membrane potential of q units (a real number between 0 and 1) which is also the initial value of the chaotic map. In general, each GLS neuron can have a different initial normalized membrane potential though in this work we assume that they are all the same. The GLS neurons have a refractory period of 1 millisecond which means that the inter-firing interval is 1 ms (from the instant they are presented with a stimulus). When the GLS neuron encounters a stimulus say x_k, the neuron starts firing chaotically until it matches the amplitude of the stimulus, i.e., when the normalized membrane potential reaches a neighbourhood of x_k, at which time it stops firing. The time duration N_k ms for which the k-th GLS neuron is active is defined as the *firing time*. The firing of each GLS neuron is guaranteed to halt (as soon as its membrane potential reaches the neighbourhood of x_k) owing to the property of Topological Transitivity (TT) which is defined below.

Definition 1 *Topological Transitivity:* Given a map $T : Q \to Q$, T is said to be topologically transitive on Q, if for every pair of non-empty open sets U and V in Q, there exist a non negative integer n and a real number $u \in U$ such that $T^n(u) \in V$.

In our example, T is the 1D GLS chaotic map with $Q : [0, 1)$. We define $U : (q - \epsilon, q + \epsilon)$ and $V : (x_k - \epsilon, x_k + \epsilon)$ as the two non-empty open sets and $\epsilon > 0$. It follows from the above definition that there exists $N_k \geq 0$ (integer) and a real number $u \in U$ such that $T^{N_k}(u) \in V$. We take $u = q$. It may be the case that certain values of q may not work, but we can always find a q that works.

A. GLS Neuron: Chaotic map

The GLS neuron [16] is a 1D map $T : [0, 1) \to [0, 1)$ defined as:

$$T(x) = \begin{cases} \frac{x}{b} & , \quad 0 \leq x < b, \\ \frac{(1-x)}{(1-b)} & , \quad b \leq x < 1 \end{cases}$$

where $x \in [0, 1)$. We have chosen $b = 0.467354$ in our study. Figure 2 represents the GLS map.

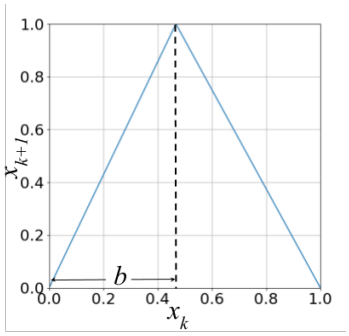

Fig. 2. The first return map of Generalized Luröth Series (GLS) [15], [16]. GLS is a chaotic 1D map that exhibits *topological transitivity*.

B. Topological Transitivity (TT) based classification algorithm

Let $\boldsymbol{X}$ be a $m \times n$ matrix where each row represents distinct data instance and the columns represent the different features. Each row (data instance) of $\boldsymbol{X}$ is $x^i = [x_1^i, x_2^i, x_3^i, ..., x_n^i] \in \mathbb{R}^n$. If the data instances are images then each x^i represents

a vectorized image with each x_k^i representing a pixel value. In our study, we have normalized[2] the values of the matrix $\boldsymbol{X}$ to lie in $[0, 1]$.

There are mainly three steps in TT based classification algorithm.

- **TT based feature extraction** - Algorithm 2 represents the TT based feature extraction.
  Let $I_k^i = (x_k^i - \epsilon, x_k^i + \epsilon)$ be the ϵ-neighbourhood of x_k^i where $\epsilon > 0$. Let N_k^i be the *firing time* of the ik-th GLS Neuron when subjected to the normalized stimulus x_k^i. This is nothing but the time in ms or equivalently the number of iterations of the GLS map T that is required to reach the interval I_k^i starting from the initial membrane potential q. In other words, $q \to T_k(q) \to T_k^2(q) \to T_k^3(q) ... \to T_k^{N_i}(q)$ where $T_k^{N_i}(q) \in I_k^i$ for the first time. The GLS neuron stops firing as soon as this is satisfied and we shall call N_k^i as *TT based feature*.
- **Training** - Algorithm 1 represents the TT based training step. Let us assume there are s classes $\mathscr{C}_1$, $\mathscr{C}_2$, ..., $\mathscr{C}_s$ with labels $1, 2, \ldots, s$ respectively. Let the data belonging to $\mathscr{C}_1$, $\mathscr{C}_2$, ..., $\mathscr{C}_s$ be denoted as $X^1, X^2, \ldots, X^s$ respectively. For simplicity, let us assume that $X^1, X^2, \ldots, X^s$ are s distinct matrices of size $m \times n$. The TT based feature extraction step is applied to $X^1, X^2, \ldots, X^s$ separately to yield $Y^1, Y^2, \ldots, Y^s$. $Y^1, Y^2, \ldots, Y^s$ have the same size as $X^1, X^2, \ldots, X^s$ since TT based feature extraction is applied to each stimulus. The average across rows is computed as follows:

$$\begin{aligned} M^1 &= \frac{1}{m}\Big[\sum_{i=1}^{m} Y_{i1}^1, \sum_{i=1}^{m} Y_{i2}^1, \ldots, \sum_{i=1}^{m} Y_{in}^1\Big], \\ M^2 &= \frac{1}{m}\Big[\sum_{i=1}^{m} Y_{i1}^2, \sum_{i=1}^{m} Y_{i2}^2, \ldots, \sum_{i=1}^{m} Y_{in}^2\Big], \\ &\vdots \\ M^s &= \frac{1}{m}\Big[\sum_{i=1}^{m} Y_{i1}^s, \sum_{i=1}^{m} Y_{i2}^s, \ldots, \sum_{i=1}^{m} Y_{in}^s\Big]. \end{aligned}$$

  These s row-vectors $M^1, M^2, \ldots, M^s$ are termed as *representation* vectors for the s classes. Each vector M^k is the *average internal representation* of all the stimuli corresponding to class k. This is biologically analogous to internal representations of experiences induced by storing *memory traces* corresponding to distinct classes in the brain.
- **Testing** - Algorithm 3 represents the testing part. Let the normalized test data be an $r \times n$ matrix denoted as Z. The ith test data instance of Z is denoted as $z^i = [z_1^i, z_2^i, z_3^i ..., z_n^i]$. TT based feature extraction is performed for each of the test data instances (z^i) where $i = 1, 2, 3, \ldots, r$. Let the resultant TT based feature extracted matrix be denoted as F, where $f^i = [f_1^i, f_2^i, \ldots, f_n^i]$ is the ith row of F. Now we compute the cosine similarity of f^i individually with each of the *representation* vectors $M^1, M^2, ..., M^s$ as follows:

$$\begin{aligned} \cos(\theta_1) &= \frac{f^i \cdot M^1}{\|f^i\|_2 \|M^1\|_2}, \\ \cos(\theta_2) &= \frac{f^i \cdot M^2}{\|f^i\|_2 \|M^2\|_2}, \\ &\vdots \\ \cos(\theta_s) &= \frac{f^i \cdot M^s}{\|f^i\|_2 \|M^s\|_2}, \end{aligned}$$

  where $\|v\|_2$ is the l_2 norm of row-vector v and $f^i \cdot M^j$ is the dot product between the row-vectors f^i and M^j. From the above s scalar values we find that index (p) which corresponds to the maximum cosine similarity between the vector M^p and f^i:

$$\theta_p = \arg\max_{\theta_i}(\cos(\theta_1), \cos(\theta_2), \ldots, \cos(\theta_s)).$$

  Now, the index p is assigned as the class label for z^i. This step is repeated until each instance of the test data is assigned a class label.

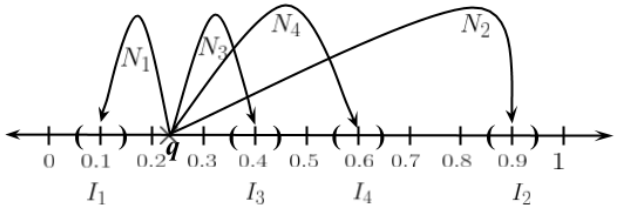

Fig. 3. Illustration of topological transitivity based feature extraction (Algorithm 2) for an example. Starting from the initial normalized membrane potential of q units, it takes N_i iterations to reach the neighbourhood I_i of the i-th stimulus.

Example: We explain the aforementioned steps with the help of an example. For simplicity, let us assume a binary classification problem with two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ with class labels 1 and 2 respectively. Let the input data be a matrix $X = \left[\frac{X^1}{X^2}\right]$ of size 4×4 where X^1 represents the first two rows of X which is the data belonging to class-1 and X^2 represents the remaining two rows of X which is the data belonging to class-2. The input layer of the proposed neuronal architecture (Figure 1) consists of 16 neurons which are denoted as $C_1, C_2, \ldots, C_{16}$. The initial membrane potential for each of these neurons is set to $q = 0.23$ units. As an example, consider the first row of X^1: $\hat{x}^1 = [1, 9, 4, 6]$. The first step is to normalize the data. After normalization, let us say we have $x^1 = [0.1, 0.9, 0.4, 0.6]$. These are the stimulus to the input layer of the GLS neural network (for the first four GLS neurons: C_1, C_2, C_3 and C_4). The stimulus initiates the firing of these 4 GLS neurons. Let us assume the *firing times* are N_1, N_2, N_3, N_4 milliseconds. As depicted in Figure 3, it takes N_k number of iterations to reach $I_k = (x_k^1 - \epsilon, x_k^1 + \epsilon)$ which is the neighbourhood of the k-th stimulus. Choosing

[2]For a non-constant matrix X, normalization is achieved by performing $\frac{X - \min(X)}{\max(X) - \min(X)}$. A constant matrix X is normalized to all ones.

$\epsilon = 0.05$, the four neighbourhoods are $I_1 = (0.05, 0.15), I_2 = (0.85, 0.95), I_3 = (0.35, 0.45), I_4 = (0.55, 0.65)$. Similary, $N_5, N_6, \ldots, N_{16}$ are the *firing times* for the GLS neurons from $C_5, C_6, \ldots, C_{16}$ respectively. This completes the TT based feature extraction step.

Algorithm 1: Training.
Input: Data-sets $\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^s$ which are s distinct $m \times n$ matrices belonging to $\mathscr{C}_1$, $\mathscr{C}_2$, ..., $\mathscr{C}_s$ respectively.
Output: Mean TT based feature extracted vectors (*representation* vectors) $M^1, M^2, \ldots, M^s$.

Normalize: $\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^s$ are normalized to $X^1, X^2, \ldots, X^s$.
Initialize: $Y^1, Y^2, \ldots, Y^s$ are each initialized to a zero matrix of size $m \times n$.
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    $Y^1_{ij} \leftarrow TT(X^1_{ij})$;
    $Y^2_{ij} \leftarrow TT(X^2_{ij})$;
    $\vdots$
    $Y^s_{ij} \leftarrow TT(X^s_{ij})$;
  end
end
$M^1 \leftarrow \frac{1}{m}\left[\sum_{i=1}^m Y^1_{i1}, \sum_{i=1}^m Y^1_{i2}, \ldots, \sum_{i=1}^m Y^1_{in}\right]$;
$M^2 \leftarrow \frac{1}{m}\left[\sum_{i=1}^m Y^2_{i1}, \sum_{i=1}^m Y^2_{i2}, \ldots, \sum_{i=1}^m Y^2_{in}\right]$;
$\vdots$
$M^s \leftarrow \frac{1}{m}\left[\sum_{i=1}^m Y^s_{i1}, \sum_{i=1}^m Y^s_{i2}, \ldots, \sum_{i=1}^m Y^s_{in}\right]$;
return $M^1, M^2, \ldots, M^s$

At the beginning of the training step, the TT based features extracted from the data are arranged as: $Y^1 = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \\ N_5 & N_6 & N_7 & N_8 \end{bmatrix}, Y^2 = \begin{bmatrix} N_9 & N_{10} & N_{11} & N_{12} \\ N_{13} & N_{14} & N_{15} & N_{16} \end{bmatrix}$. In the training step, we compute the two *representation* vectors corresponding to the two classes as $M^1 = \frac{1}{2}[N_1 + N_5, N_2 + N_6, N_3 + N_7, N_4 + N_8]$ and $M^2 = \frac{1}{2}[N_9 + N_{13}, N_{10} + N_{14}, N_{11} + N_{15}, N_{12} + N_{16}]$.

Once the *representation* vectors are computed, we are ready to perform the testing on unseen data. Assume that the test data is a matrix Z of size 2×4. We are required to classify each row of Z to belong to either of class $\mathscr{C}_1$ or $\mathscr{C}_2$. We first normalize the matrix Z so that it contains only real values between 0 and 1. The TT based features are extracted for Z by recording the firing times of the GLS neurons to yield $F = \begin{bmatrix} f^1_1 & f^1_2 & f^1_3 & f^1_4 \\ f^2_5 & f^2_6 & f^2_7 & f^2_8 \end{bmatrix}$. In order to classify f^1, the first row of F (and hence the first row of Z), we compute the cosine similarity measure between f^1 and the two representation vectors M^1 and M^2 independently (say $\cos(\theta_1)$ and $\cos(\theta_2)$). We find the maximum of these two values, say $\cos(\theta_2)$. In this case, the label assigned to the first row of Z would be 2. We repeat this for the second row of F. In this way, the unseen test data is classified using the representation vectors.

Algorithm 2: TT based feature extraction.
Input: Stimulus (normalized) x to the GLS neuron (chaotic map $T(\cdot)$).
Output: *Firing time* N.

Initialize: GLS neuron is initialized with membrane potential q, ϵ-neighbourhood $I = (x - \epsilon, x + \epsilon)$ with $\epsilon = 0.1$.
$w \leftarrow q$;
$N \leftarrow 0$;
while $(w \in I == False)$ **do**
  $w \leftarrow T(w)$;
  $N \leftarrow N + 1$;
end
return N

Algorithm 3: Testing.
Input: Test-data $\hat{Z}$ of size $r \times n$ and *representation* vectors $M^1, M^2, \ldots, M^s$.
Output: Predicted labels - L.

Normalize: $\hat{Z}$ is normalized to Z.
Initialize: L is initialized as a zero vector of size $r \times 1$.
for $i \leftarrow 1$ **to** r **do**
  for $j \leftarrow 1$ **to** n **do**
    $F_{ij} \leftarrow TT(Z_{ij})$;
  end
end
for $i \leftarrow 1$ **to** r **do**
  $\cos(\theta_1) \leftarrow cosinesimilarity(f^i, M^1)$;
  $\cos(\theta_2) \leftarrow cosinesimilarity(f^i, M^2)$;
  $\vdots$
  $\cos(\theta_s) \leftarrow cosinesimilarity(f^i, M^s)$;
  $\theta_p \leftarrow \arg\max_{\theta_k}(\cos(\theta_1), \cos(\theta_2), \ldots, \cos(\theta_s))$;
  $L_{i1} \leftarrow p$;
end
return L

Algorithm 4: Cosine similarity computation.
Input: Vectors $u = [a_1, a_2, \ldots a_n]$ and $v = [b_1, b_2, \ldots b_n]$.
Output: Cosine similarity measure $\cos(\theta)$.

$\cos(\theta) \leftarrow \frac{a_1 b_1 + a_2 b_2 + \ldots + a_n b_n}{\sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}\sqrt{b_1^2 + b_2^2 + \ldots + b_n^2}}$;
return $\cos(\theta)$

C. Hyperparameters

The hyperparameters used in this method are as follows:

1) Map and its properties: In the proposed algorithm, we used 1D GLS chaotic map for the neuron. In general, we can also use other chaotic maps (such as logistic map) that satisfy the topological transitivity property. In the GLS map used in the proposed method (Figure 2), b is another hyperparameter.

2) The initial normalized membrane potential q, which is also the initial value for the chaotic map, is another hyperparameter. This initial value can be different for each GLS neuron, though in our work we have chosen the same for all the neurons.

The above hyperparameters can be tuned to further improve the performance.

III. Experiments and Results

Learning from limited samples is a challenging problem in the AI community. We evaluate the performance of our proposed TT based classification on MNIST [12] and Iris data[3] [17] with limited training data samples. A brief description of these datasets is given below.

A. Datasets

1) MNIST: This dataset consists of hand written digits from 0 to 9 stored as digitized 8-bit grayscale images with dimensions of 28 pixels $\times$ 28 pixels with a total of 60000 images for training and 10000 images for testing. This is a 10-class classification task, i.e., the goal is to automatically classify these images to the ten classes corresponding to the digits 0 to 9. In our study, we performed independent trials of training with only $1, 2, 3, \ldots, 20, 21$ data samples per class (randomly chosen from the available 60000 training images). For each trial, we tested our algorithm with 10000 unseen test images.

2) Iris data: This dataset consists of 4 attributes of 3 types (classes) of Iris plants (namely Setosa, Versicolour and Virginica). The 4 attributes are: sepal length (cm), sepal width (cm), petal length (cm) and petal width (cm). There are 50 data samples per class. This is a 3-class classification problem. In this study, we performed independent trials of training with randomly chosen $1, 2, 3, \ldots, 5$ data samples per class. For each trail, we tested with 120 unseen randomly chosen data samples.

B. Comparative performance evaluation of the proposed method with other methods

We compare our method with existing algorithms in literature. Specifically, we compare with Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbour (KNN) and 2-layer neural network. The machine learning techniques used in this research are implemented using *Scikit-learn* [18]. The default parameters in *Scikit-learn* for DT, SVM and KNN are used in this research. We have used *gini* criterion for DT classifier and *radial basis function* (RBF) kernel for SVM based classification. For KNN, the number of nearest neighbours used was 5. We have used *Keras* [19] package for the implementation of 2-layer neural network with 784 neurons in the input layer, 784 neurons in the hidden layer and 10 neurons in the output layer for MNIST classification task. For Iris data classification task, we used 4 neurons in the input layer, 4 neurons in the hidden layer and 3 neurons in the output layer.

[3]http://archive.ics.uci.edu/ml/datasets/iris

The comparative performance of TT based method and ML methods for MNIST and Iris data are provided in Figure 4 and 5 respectively. From these results, we make the following observations:

- The proposed method shows consistent performance in the low training sample regime for both datasets.
- For the MNIST dataset, our method outperforms the classical ML techniques - SVM, KNN, and DT. When compared with DL (2-layers) the method closely matches the accuracy up to 7 training samples/class.
- For the Iris dataset, our method has the best performance when trained with 2 samples/class. DL (2-layers) gave the least accuracy throughout the low training sample regime.

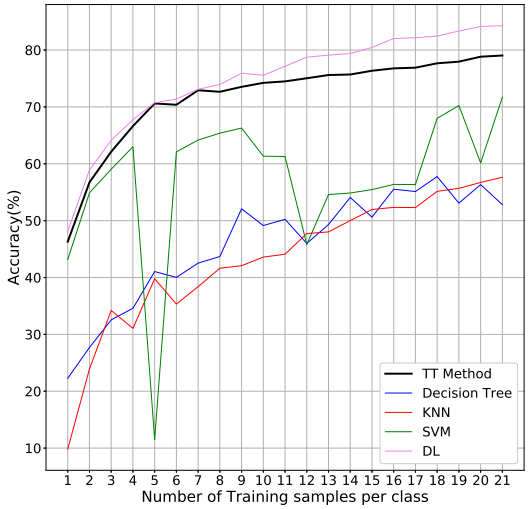

Fig. 4. Comparative performance evaluation of TT based method with DT, SVM, KNN and DL (2-layers) for MNIST dataset in the low training sample regime.

IV. Conclusions and Future Research Directions

As evident from the results, TT based classification gives a consistent performance in low sample regime compared to classical ML/DL techniques. This method can be particularly useful when the number of available training samples is less. As the size of training data increases, conventional ML/DL methods outperform our method. A direction for future research is to investigate whether we can combine TT based method with ML/DL methods to yield a superior hybrid algorithm.

A significant advantage of the proposed method is that it need not be re-trained from scratch if a new class (with new data samples) is added. The *representation* vectors of all the existing classes will not change. Only the representation vector for the new class needs to be computed. In contrast, such a scenario would require a complete re-designing and re-training in the case of ML and DL.

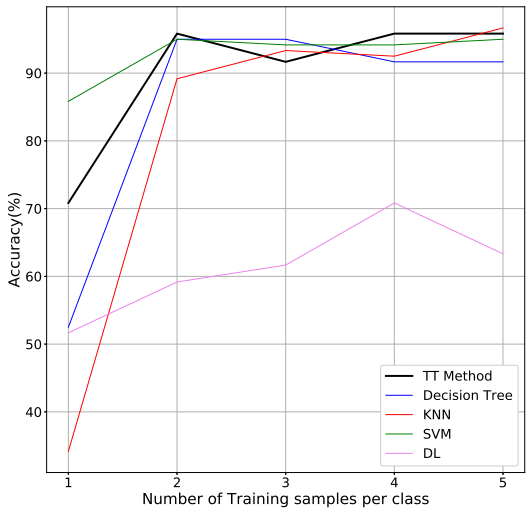

Fig. 5. Comparative performance Comparison of TT based method with DT, SVM, KNN and DL (2-layers) for Iris dataset in the low training sample regime.

Our method has fewer hyperparameters than ML/DL methods. It can be noticed that we have not performed hyperparameter tuning in this work since we are dealing with very few training samples. Future work could involve using multiple chaotic maps (such as logistic map) and also designing a network with multiple layers of chaotic neurons. We expect that such modifications can further increase the accuracy.

To conclude, we have for the first time proposed a novel chaos based neural architecture which makes use of the property of *topological transitivity*. In our architecture, the non-linearity and chaos is intrinsic to the neuron unlike conventional ANN. Earlier research ([20] and [21]) highlight the presence of neurons in the hippocampus (of the rat's brain) which are sensitive to a particular point in space. In a similar vein, our method proposes *temporally* sensitive neurons. In the proposed model, the *firing time* of the chaotic GLS neuron required to match the response of the stimulus is a discriminating feature to distinguish different classes. Thus, our research is an initial step towards employing chaos (and its fascinating properties) in an intrinsic fashion to design novel learning architectures and algorithms that are inspired from the biological brain.

Acknowledgment

H.N.B. thanks "The University of Trans-Disciplinary Health Sciences and Technology (TDU)" for permitting this research as part of the PhD program. The authors gratefully acknowledge the financial support of Tata Trusts. We dedicate this work to late Prof. Prabhakar G Vaidya who initiated us to the fascinating field of Chaos Theory.

References

[1] Vilayanur S Ramachandran, Sandra Blakeslee, and Neil Shah. *Phantoms in the brain: Probing the mysteries of the human mind*. William Morrow New York, 1998.

[2] Frederico AC Azevedo, Ludmila RB Carvalho, Lea T Grinberg, José Marcelo Farfel, Renata EL Ferretti, Renata EP Leite, Wilson Jacob Filho, Roberto Lent, and Suzana Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009.

[3] Philippe Faure and Henri Korn. Is there chaos in the brain? i. concepts of nonlinear dynamics and methods of investigation. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, 324(9):773–793, 2001.

[4] Henri Korn and Philippe Faure. Is there chaos in the brain? ii. experimental evidence and related models. *Comptes rendus biologies*, 326(9):787–840, 2003.

[5] Gabriela Czanner, Sridevi V Sarma, Demba Ba, Uri T Eden, Wei Wu, Emad Eskandar, Hubert H Lim, Simona Temereanca, Wendy A Suzuki, and Emery N Brown. Measuring the signal-to-noise ratio of a neuron. *Proceedings of the National Academy of Sciences*, 112(23):7141–7146, 2015.

[6] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[7] NB Harikrishnan, R Vinayakumar, and KP Soman. A machine learning approach towards phishing email detection. In *Proceedings of the Anti-Phishing Pilot at ACM International Workshop on Security and Privacy Analytics (IWSPA AP). CEUR-WS. org*, volume 2013, pages 455–468, 2018.

[8] Yiming Ding, Jae Ho Sohn, Michael G Kawczynski, Hari Trivedi, Roy Harnish, Nathaniel W Jenkins, Dmytro Lituiev, Timothy P Copeland, Mariam S Aboian, Carina Mari Aparici, et al. A deep learning model to predict a diagnosis of alzheimer disease by using 18f-fdg pet of the brain. *Radiology*, 290(2):456–464, 2018.

[9] A Zerroug, L Terrissa, and A Faure. Chaotic dynamical behavior of recurrent neural network. *Annu. Rev. Chaos Theory Bifurc. Dyn. Syst*, 4:55–66, 2013.

[10] JC Sprott. Is chaos good for learning? *Nonlinear dynamics, psychology, and life sciences*, 17(2):223–232, 2013.

[11] Charles B Delahunt and J Nathan Kutz. Putting a bug in ML: The moth olfactory network learns to read MNIST. *arXiv preprint arXiv:1802.05405*, 2018.

[12] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[13] Aditi Kathpalia and Nithin Nagaraj. A novel compression based neuronal architecture for memory encoding. In *Proceedings of the 20th International Conference on Distributed Computing and Networking*, pages 365–370. ACM, 2019.

[14] Nithin Nagaraj. *Novel applications of chaos theory to coding and cryptography*. PhD thesis, NIAS, 2008.

[15] Nithin Nagaraj, Prabhakar G Vaidya, and Kishor G Bhat. Arithmetic coding as a non-linear dynamical system. *Communications in Nonlinear Science and Numerical Simulation*, 14(4):1013–1020, 2009.

[16] Karma Dajani and Cor Kraaikamp. *Ergodic theory of numbers*. Number 29. Cambridge University Press, 2002.

[17] Catherine L Blake and Christopher J Merz. UCI repository of machine learning databases, 1998.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] François Chollet et al. Keras. https://keras.io, 2015.

[20] John O'Keefe. Place units in the hippocampus of the freely moving rat. *Experimental neurology*, 51(1):78–109, 1976.

[21] John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 1971.