Machine Learning with Applications 6 (2021) 100080

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Comparative analysis of credit card fraud detection in Simulated Annealing trained Artificial Neural Network and Hierarchical Temporal Memory

E.N. Osegi [a,*], E.F. Jumbo [b]

[a] Department of Information Technology, National Open University of Nigeria (NOUN), Lagos, Nigeria
[b] Department of Computer Sciences, University of Port Harcourt, Choba, Nigeria

ARTICLE INFO

Keywords:
Hierarchical Temporal Memory
Artificial Neural Network
Simulated Annealing Algorithm
Cortical Learning Algorithm
Misclassification
Sparse distributed representation

ABSTRACT

The problem of misclassification has always been a major concern in detecting online credit card fraud in e-commerce systems. This concern greatly poses a significant challenge to financial institutions and online merchants with regards to financial loss. This paper specifically compares an Artificial Neural Network trained by the Simulated Annealing technique (SA-ANN) with a proposed emerging online learning technology in anomaly detection known as the Hierarchical Temporal Memory based on the Cortical Learning Algorithms (HTM-CLA). Comparisons are also made with a deep recurrent neural technique based on the Long Short-Term Memory ANN (LSTM-ANN). The performances of these systems are investigated on the basis of correctly classifying credit card fraud (CCF) using an average classification performance ratio metric. The results of simulations on two CCF benchmark datasets (the Australian and German CCF data) showed promising competitive performance of the proposed HTM-CLA with the SA-ANN. The HTM-CLA also clearly outperformed the LSTM-ANN in the considered benchmark datasets by a factor of 2:1.

1. Introduction

Credit card fraud remains a major challenge to most e-commerce systems due to its direct (personal) to indirect (non-personal) financial effect on the potential victim. The implication is heavy losses on the side of the legitimate credit card holder (CCH). Some of the direct effects are due to CCH negligence of security breaches or just carelessness while some indirect effects are due to merchants e.g. overcharging and non-delivery of purchased items (Falaki, Alese, & Ismaila, 2010). Other issues include the vulnerability in the credit card device itself or through online (internet) access e.g. by way of Card Not Present (CNP) feature (Budhram, 2012; Falaki et al., 2010). Hence, it is still a big challenge for merchants to confirm the genuineness of the customer carrying out a transaction. Machine learning algorithms inspired by artificial intelligence principles come in handy in this regard. These algorithms have been widely used by different industries and are currently being used by financial fraud detection experts to enhance the classification accuracy of existing or more traditional statistical classifiers such as the simple regressors.

The ANN so far is one of the few techniques in machine learning that has recorded a high degree of accuracy, however, the recent online machine learning model, HTM which is based on the Cortical Learning Algorithms (HTM-CLA), developed earlier in Hawkins, Ahmad, and Dubinsky (2011) has a structure that promises better intelligent prediction and possibly classification particularly with respect to missing labels and small samples of sequentially programmed data.

Thus, the primary research objective is to validate this very important emergentist neural network scheme for the task of Credit Card Fraud (CCF) detection. It is also the objective of this research to compare the HTM-CLA with the widely used Artificial Neural Network (ANN) trained by a very popular and fundamental heuristic optimizer — the Simulated Annealing (SA) technique developed in Kirkpatrick, Gelatt, and Vecchi (1983) and the deep recurrent ANN model called the Long Short-Term Memory (LSTM) developed earlier in Hochreiter, and Schmidhuber (1997). The SA technique follows from important physical phenomena (in this case temperature conditioning of metals) as in the metallurgical sciences and is very simple to implement. It also has very good convergence response avoiding the local optimality issue using the progressive refinement approach (Shahookar, & Mazumder, 1991). On the other hand, the LSTM technique uses a recurrent gating structure to effectively learn multiple layers of representations while avoiding the pitfall of vanishing gradients (Anireh, & Osegi, 2017; Osegi, & Anireh, 2020).

This research works focuses on the HTM technologies developed for predictive applications. Artificial Neural Networks (ANNs) are very

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: emmaosegi@gmail.com (E.N. Osegi), evans3447@gmail.com (E.F. Jumbo).

https://doi.org/10.1016/j.mlwa.2021.100080
Received 10 October 2020; Received in revised form 15 June 2021; Accepted 16 June 2021
Available online 20 June 2021

successful in a large number of classification, regression and prediction tasks. However, one major challenge faced by such conventional neural network strategies is the problem of dealing with long transitions of temporally rich data — a key feature of credit card transactions. As an additional problem, continual (learning) is virtually absent in most standard ANNs. Thus, it is desirable to develop techniques that are compatible with real time processing i.e. systems that can handle continual streaming data sources.

In this research, we make the following important contributions:

i. The comparative validation of a continually learning technique, HTM-CLA based on the concept of average classification performance ratio (ACPR) on two very popular and widely used CCF benchmark datasets — The German and Australian CCF datasets.
ii. The development of a new encoder scheme — the Reference Equivalency Matching (REM) for the encoding of raw input feature data into integer representations and binary SDRs.

In subsequent section (Section 2) we present some existing works that have been used in CCF detection or classification. Cortical Learning Algorithms (CLA) in the context of Hierarchical Temporal Memory (HTM) describing some very important topologies used in different HTM-like applications and how the HTM solves the anomaly detection problem are presented in Section 3. Section 4 presents the techniques used for comparative analysis; these techniques include the Simulated Annealing based ANN (SA-ANN) and the Long Short-Term Memory ANN (LSTM-ANN) schemes.

Section 5 presents the experimental results using HTM-CLA neural technique on a German and Australian credit card dataset benchmarks and a discussion of results. A comparison with the SA-ANN and LSTM techniques are carried out based on the ACPR concept. The practical implications of the research study and the novel encoder scheme used are also presented in this section.

Finally, we give our conclusions and future research directions for the proposed HTM-CLA technique.

2. Background and related work

The use of artificial neural networks for anomaly detection such as credit card fraud detection has been investigated by several researchers. Some of these researches include for instance in Patidar, and Sharma (2011) where the Genetic Algorithm (GA) trained ANN is used, in Singh, Karnwal, Prasad, D'Souza, and Shenoy (2006) where a standard multilayered ANN is trained on synthetic data, in Kamaruddin, and Ravi (2016) where an auto-associative neural network is trained with a Particle Swarm Optimizer (PSO) technique, in Wang et al. (2018) where a Whale Optimization (WO) trained ANN is used and in Zou, Zhang, and Jiang (2019) where denoising auto-encoder neural network (DAE) is used for classifying CCF data from Kaggle website. A comprehensive discussion of these investigations can be found in Chandola, Banerjee, and Kumar (2009) and Falaki et al. (2010).

HTM-CLA is an emerging neural scheme that offers an alternative artificial neural network solution to the Temporal CCF detection problem. HTM-CLA is a biological plausible algorithm that attempts to replicate the operations of the neocortex in computer software algorithm. As a machine learning framework, it primarily employs a spatio-temporal learning algorithm to form invariant features of the input world. This feature processing phase is supported by several software algorithms, techniques, tools and devices. Thus, HTM is not an entirely novel AI inspired technology, but a marriage of existing and evolving computational technologies (Hawkins, Ahmad, Purdy, & Lavin, 2016).

One very important advantage of HTM-CLA over conventional neural schemes is in its ability to handle continual streams of data sources. In this research, we propose the HTM-CLA as a candidate tool/technique for CCF detection and classification.

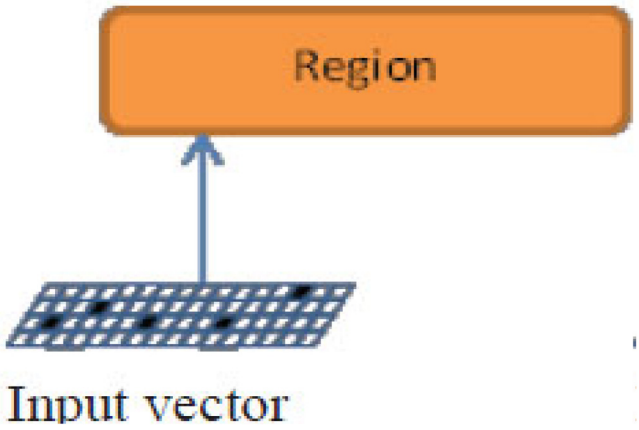

Fig. 1. Single-Layer HTM-CLA. The input vector is typically a sparse representation (black dots) of real world inputs but this is not always constrained to be so. The region represents a group of columns and cells that are dynamically generated using a spatial–temporal pooling stage.
Source: Agrawal, and Franklin (2014).

3. HTM-CLA architectures and the anomaly detection problem

In this section we review Cortical Learning Algorithm (CLA) used for online and unsupervised learning — this is the primary learning algorithm based on Hierarchical Temporal Memory (HTM). Such systems are useful for anomaly detection. We shall refer to this class of computational algorithms as HTM-CLAs.

Several architectures of HTM-CLAs exist and are currently being investigated by some machine learning researchers. These architectures are a variation of the original HTM-CLA model introduced in Hawkins et al. (2011) and include standard Single-Layer HTM-CLA, Predictive Coding (PC) HTM-CLA, Multi-Layer HTM-CLA and the Temporal-Aggregated Learning HTM-CLA (TA-HTM-CLA). The topologies are briefly described in the following sub-section (Sections 3.1, 3.2 and 3.4). Also, how the HTM-CLA technique solves the problem of classification and hence anomaly detection or prediction is introduced in Section 3.5.

3.1. Single-Layer HTM-CLA

Single-Layer HTM-CLA architecture was proposed earlier in Hawkins et al. (2011) and is shown in Fig. 1. It is loosely based on the memory prediction framework earlier conceptualized in Hawkins, and Blakeslee (2004).

It includes the desirable properties of sparse-distributed representations, hierarchical spatial–temporal processing and large synapse count. Some key learning routines employed in the spatial pooling phase include the concept of overlap, local activity, boosting, duty cycle and k-score rating. In addition the temporal phase operates at a somewhat meta-level and includes the core functional routines of segment activation, adaptation, and matching.

Learning in HTM-CLA uses standard Hebbian updates and the notion of permanence - a term used to describe the threshold connectivity level of the input synapses. This approach to machine learning adds noise robustness and full online feature learning capability to data mining applications resulting in considerable improvements in prediction accuracies.

3.2. PC HTM-CLA

Introduced in McCall, and Franklin (2013), this architecture adds predictive coding capabilities to the Single-Layer CLA. Its structural outlay is depicted in Fig. 2 and includes the processing of hierarchical

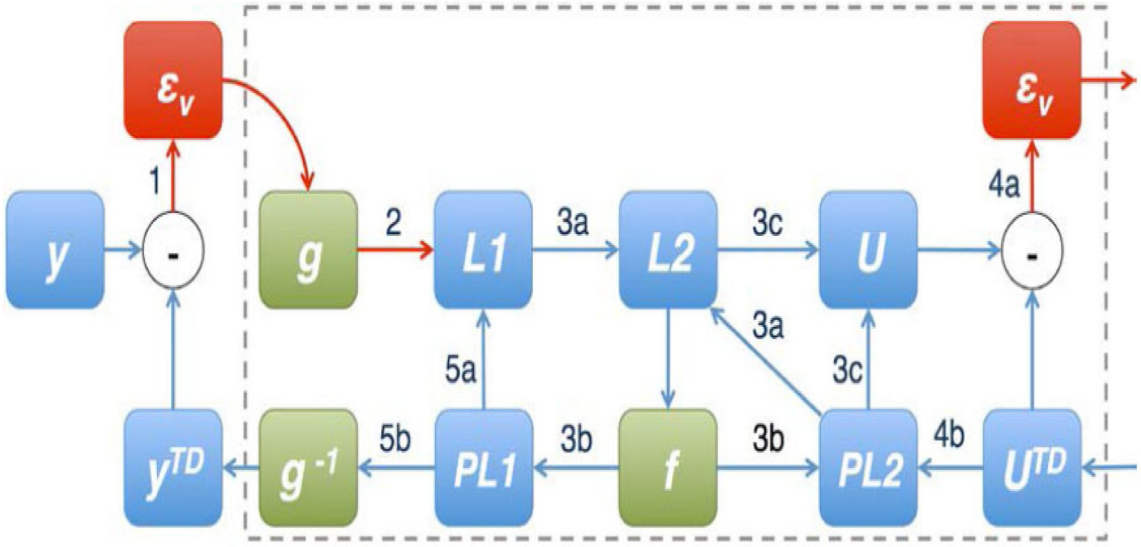

Fig. 2. A Predictive-Coding (PC) HTM-CLA. The notations g, g^{-1}, and f represents a set of feed-forward connection weights, feedback connection weights and cortical region process respectively. PL1 and PL2 are cells predicted to be active at next and future cycles. U is the union of U at PL2, U^{TD} the current received top-down prediction, and ϵ_v, the current bottom-up prediction error between a current bottom-up Boolean input, y and a previous cycle top-down prediction, y^{TD}. 1, 2, 3a–c, 4a–b, 5a–b are predictive coding steps.
Source: McCall and Franklin (2013).

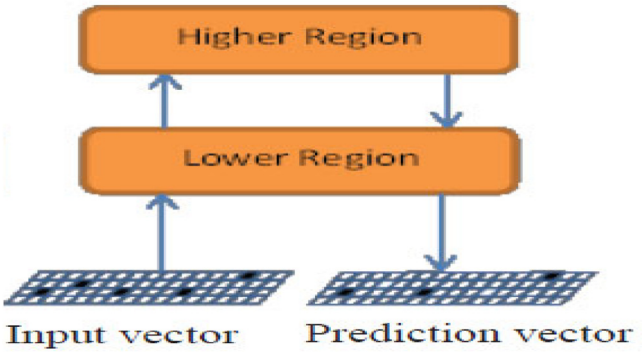

Fig. 3. A Two-Layer HTM-CLA; here the lower region communicates with a higher region using meta-feed-forward and feedback processing to form a prediction vector.
Source: Agrawal and Franklin (2014).

cortical regions. The primary purpose of this topology is to minimize the prediction errors by evaluating the top-down predictions with respect to the bottom-up feed-forward input at the previous time step.

3.3. Multi-layer HTM-CLA

This architecture is simply an extension of the Single-Layer HTM-CLA by the addition of more cortical regions. Specifically, a Multi-Layer (ML) HTM-CLA is a three-level hierarchy system consisting of an input-level and two CLA regions (Agrawal & Franklin, 2014). ML HTM-CLA is illustrated in Fig. 3.

3.4. Temporal-aggregated (TA) HTM-CLA

This is a standard HTM-CLA system based on the HTM Spatial Pooling (SP) model introduced earlier in Cui, Ahmad, and Hawkins (2017). It is specifically targeted at real time continual learning applications (Osegi, 2021). The primary idea behind the TA-HTM-CLA is to adaptively build a cause and effect model from a spatial-pooled univariate time series using a temporal bivariate sequencer. In a TA-HTM-CLA emphasis is placed more on temporal causality rather than simply spatial–temporal completeness. The systems level architecture of a TA-HTM-CLA is as shown in Fig. 4.

3.5. The classification problem as an anomaly detection problem

The classification problem typically defines a situation in which we seek to discover or identify a discrepancy between a normal observation and that which is abnormal. Thus, this kind of task can be defined as an Anomaly Detection Problem (ADP). Due to the ever changing requirements of real time streaming data processing, systems that possess online learning capabilities are highly favored. Supervised learning systems such as feed-forward back-propagation neural networks are useful in anomaly classification tasks and have been widely applied in practice. However, the extra requirement of online processing makes them less attractive in fraud detection systems (Galetzka, 2014; Hawkins, Ahmad, & Dubinsky, 2014). Unsupervised learning systems such as HTM-CLA are able to overcome this drawback and are capable of learning through indirect supervision. How this is possible is illustrated in the anomaly detection model shown in Fig. 5.

In neural network terms, a learning algorithm tries to find the set of weights and biases that minimizes the objective, in this case the sum of square error loss. While this is so for most neural networks, HTM-CLA uses a different metric for evaluating its performance — the overlap principle. The overlap is basically the sum over the logical product of its generative binary vectors compared with the input vector. The overlap concept uses this notion to select cortical columns that likely meet an overlap criterion, while ignoring the rest (Ahmad, & Hawkins, 2015; Cui et al., 2017; Osegi, 2021). This inhibition process results in a self-tuned predictive system. It is important to note that these operations take place in a sequential feed-forward fashion with respect to time. So the challenge is actually to interpret the meaning of performance when trying to compare a supervised and an unsupervised learning network. One attempt at this is to use unsupervised training to evolve the model, then apply prefix or beam search for novel inputs on the evolving model.

4. Overview of the comparative techniques

The additional techniques considered in this research are presented in this section; the Simulated Annealing based ANN (SA-ANN) and

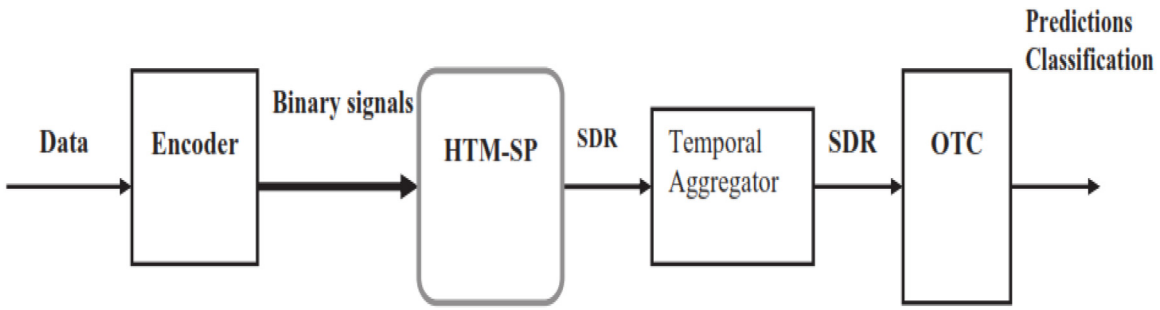

Fig. 4. A TA- HTM-CLA.
Source: Osegi (2021).

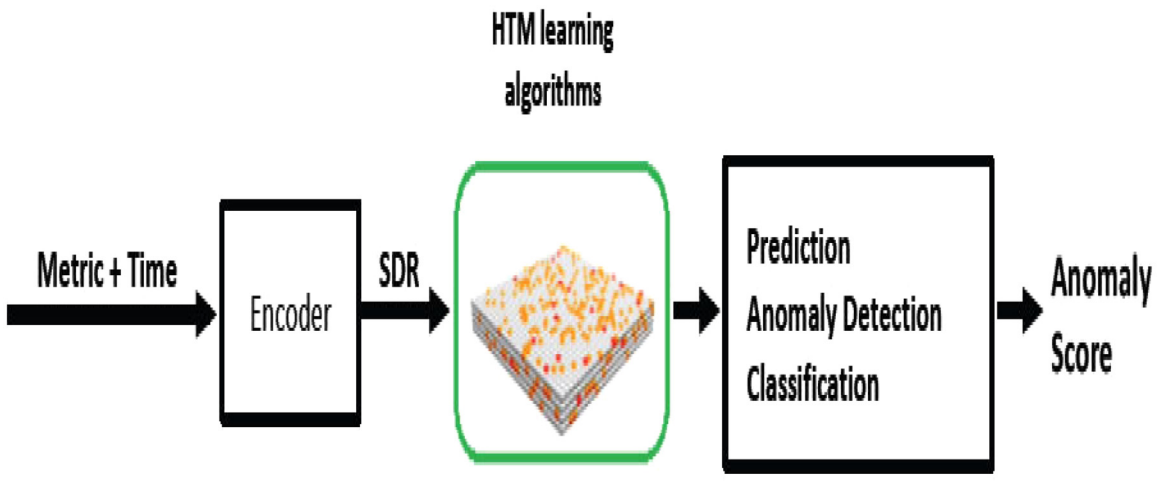

Fig. 5. Anomaly detection model.
Source: Hawkins et al. (2014).

a deep recurrent neural network called Long Short-Term Memory (LSTM). These techniques are described succinctly in the sub-sections that follow.

4.1. The simulated annealing artificial neural network

The Simulated Annealing Artificial Neural Network (SA-ANN) uses the concept of thermal conditioning (heating and cooling) to improve a standard ANN learning performance on the input training data. This idea of thermal conditioning follows from the research earlier proposed in Kirkpatrick et al. (1983). This idea is further substantiated from the Metropolis process which models the transition of heated solids at equilibrium states from a set of initial states (Sexton, Dorsey, & Johnson, 1999).

Specifically, the simulated annealing (SA) process follows a repeated set of iterative refinements (uphill or downhill moves) to find a good solution. In an SA-ANN, a solution vector say V, with its corresponding weights, W, is obtained by minimizing a cost function, say f – typically the sum of squared errors (SSE). This process involves the determination of candidate solutions from an initial random solution set based on the model in Eq. (1):

$$w_i' = w_i + r * v_i \tag{1}$$

where,
w_i = weight vector.
w_i' = candidate solution weight obtained by varying w_i.
v_i = step length of w_i.
r = random number.
Using the new weight computed in Eq. (1), the corresponding new fitness, f' is computed and if found less than an initial pre-computed value f, its value and the associated w_i value are saved as best solutions. Step lengths are also adjusted accordingly such that 50% of all moves are preserved.

Otherwise, the Metropolis acceptability criterion is used as in Eq. (2):

$$p = e^{\frac{(f-f')}{T}} > p' \tag{2}$$

where,
p = a uniform user defined random number between 0 and 1.
T = annealing temperature.
Now, if p is greater than 0, the new solution point is accepted; otherwise the solution is rejected and the simulation repeated. This process continues until the desired solution level is met or the total number of fixed simulation trials is attained.

The adjustment (reduction) in temperature, T, is computed as in Eq. (3) as:

$$T_1 = r_1 * T_0 \tag{3}$$

where,
r_1 = a uniform user defined random number between 0 and 1.
T = annealing temperature.
If T is reduced, the probability of accepting uphill moves in the solution process will be reduced. The converse is also valid. Thus, this gives diversity in the solution process and enables the SA escape local optima states.

4.2. The Long Short-Term Memory artificial neural network

Long Short-Term Memory (LSTM) neural networks are recurrent learning techniques used to solve a variety of sequence learning problems. The LSTM was introduced earlier in Hochreiter and Schmidhuber

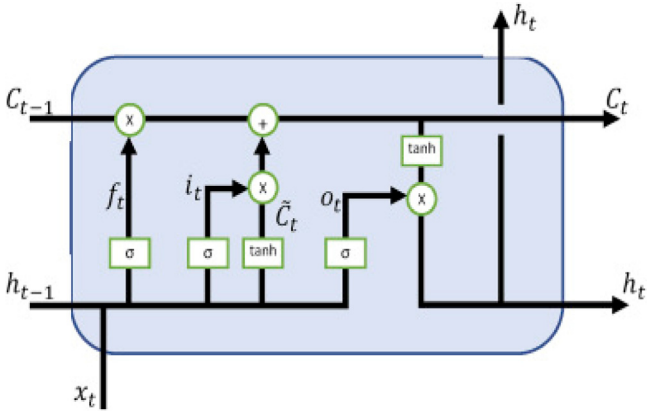

Fig. 6. LSTM neuron or cell. *Source:* Hochreiter and Schmidhuber (1997).

(1997). Their superiority lies in their ability to effectively overcome the vanishing or exploding gradient problems common in other conventional backpropagation trained neural networks. This also allows for network to remember long term dependencies. A common network consists of input, output and a number of hidden layers as in conventional neural networks; however, an LSTM neuron uses a special neuron called a cell as shown in Fig. 6.

In this architecture, x represents an input vector, h denotes its cell output and C denotes the corresponding cell memory. For each time step, four vectors are calculated by concatenating dot products of previous hidden states and input vectors. Also, f represents the forget gate, i the input gate, $\tilde{c}$ the candidate gate and o is the output gate.

In an LSTM cell, forget gate is applied to filter only redundant parameters and long-term dependencies are passed through unaffected. This allows for required memory state to be transmitted to next cell unaffected and thus gradients are controlled. Training and inference is then effectively done by unrolling the LSTM network in time (Fayek, 2019). Thus, LSTM neural networks can effectively make long range predictions without getting stuck or encountering long delays in the process.

5. Experimental results and discussions

Real world observations are generally temporal in nature possessing the unique property of system inertia, recurrence and ocular dominance. For instance, certain tasks have the requirement that the observation(s) vary slowly with time or remain stationary while some tasks demand that frequent observations be preferred over less frequent ones and vice versa. When an observation is partially observed, learning spatial representations is difficult — this is synonymous with the blindfold principle with one eye dominating over the other. However, this becomes less difficult with a temporal representation. The classification task presented here will test the abilities of a standard feed forward multi-layer perceptron ANN trained with simulated annealing, the HTM-CLA network adapted from Hawkins et al. (2011) that follows from the blindfold principle and the LSTM deep recurrent technique. For the purposes of this study, we consider the case where the data to be analyzed is spatial in nature.

The classification task domain uses sample data from the German credit card fraud benchmark dataset defined in Hofmann (1996) and the dataset from the Australian credit card fraud benchmark (Quinlan, 1987, 1992). The attributes of these datasets are provided in Appendix A; full details of these datasets can be obtained from the UCI repository at https://archive.ics.uci.edu.

5.1. Experimental setup

The HTM-CLA system uses the original spatial pooler algorithm as discussed in Section 3, Section 3.1. The temporal pooling (temporal-memory) stage is based on the ideas proposed in Ahmad and Hawkins (2015). The temporal memory (T-M) stage follows the spatial-pooler (SP) so that sufficient invariant representations can be formed. The SP and TM stage captures some desirable properties in cortical learning microcircuits including but not limited to slowness of time, hierarchical representation of recognition states, a generative belief propagation, probabilistic sharing, causal and transfer learning. These stages are illustrated in Fig. 7. The TM stage may be omitted from system architecture without a drastic reduction in classification accuracy, however we do use this stage as a proof-of concept that HTM-CLA works accordingly.

In particular we use a maximizing temporal recurrent index operation to get the predicted spatial pattern following an overlapping rule based on previously formulated SP SDRs and novel SDRs.

In order to demonstrate HTM-CLA's unique ability to improve over time, we vary the number of columns by fixing a few set of hyper-parameters at specific (default) values (see Appendix B). The primary parameters of interest are the desired local activity, number of iterations or runs, permanence threshold and minimum overlap. The training is done online (continually) with 70% of the data used as online training data while the remaining 30% for online prediction. For both considered datasets and from initial experiments, we have defined the initial permanence and stimulus thresholds for the HTM-CLA as in Appendix B (Table B.1).

For the SA-ANN part, we optimize the neural network learning phase by evolving the number of hidden neurons with an annealing operation. The hidden neuron is a crucial parameter in a multi-layer ANN due to its feature detecting capabilities (García-Pedrajas, Ortiz-Boyer, & Hervás-Martínez, 2006). The testing is done on 30% of training data to verify the SA-ANN model performance accuracy.

The LSTM-ANN uses several key hyper-parameters which are fine tuned to give a reasonable performance level. The size of the hidden layer is fixed at 4×4 while the batch size is varied depending on the considered CCF dataset (see Table B.3); for each considered sample size (as per the German CCF dataset), the learning rate is adjusted as in Table B.4 (see Appendix B).

In the both CCF dataset experiments, it is assumed that the testing data has not been seen before by the classifier techniques. Thus, a correct prediction is attained if the prediction matches the class label assigned to the training example captured during testing. To enforce this feature we miss the labels in the testing data of both techniques i.e. we obscure the class label values for the test examples. Technically, this is achieved by replacing the actual testing data class label values with character 49 and 48 — the equivalent of number 1 or 0 respectively in the ASCII character set. Experimental parameters for the conducted experiments for the HTM-CLA, SA-ANN and LSTM-ANN are provided in Appendix B. In the sub-sections that follow (Sections 5.2 and 5.3), we describe the results obtained from the CCF classification experiment.

5.2. Results using the German credit card benchmark

The classification performance of the considered techniques obtained using the parameters defined in Appendix B and sample German CCF data provided in Hofmann (1996) are as presented in Tables 1–3 for the proposed HTM-CLA, the SA-ANN and LSTM-ANN techniques respectively. We use an exact equivalency matching scheme to estimate the classification (or misclassification) rate for each target exemplar on the test set. For the LSTM-ANN, further learning rate parameter tuning as per the considered sample sizes are performed in accordance to Table B.3 (see Appendix B). Also, due to poor initial classifications, the epochs (number of training iterations) of the LSTM-ANN was set at

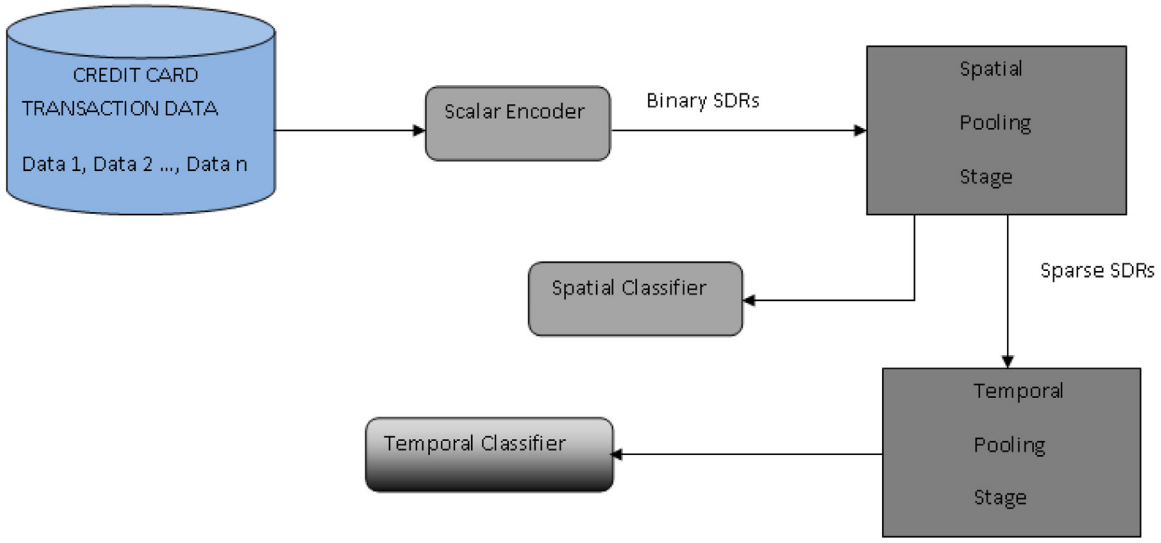

Fig. 7. Architecture of the proposed HTM-CLA classifier system.

Table 1
Classification performance of the HTM-CLA; the subscripts denote number of samples.

s/n	HTM_{50}	HTM_{100}	HTM_{150}	HTM_{200}	HTM_{250}	HTM_{300}
1	80.00	80.00	24.44	73.33	66.67	28.89
2	80.00	80.00	75.56	73.33	50.67	71.11
3	80.00	80.00	46.67	73.33	66.67	71.11
4	80.00	80.00	75.56	73.33	66.67	71.11
5	20.00	20.00	75.56	73.33	66.67	71.11
6	80.00	80.00	75.56	73.33	66.67	71.11
7	80.00	80.00	75.56	73.33	66.67	71.11
8	80.00	80.00	75.56	73.33	66.67	58.89
9	80.00	80.00	75.56	73.33	66.67	71.11
10	20.00	80.00	75.56	73.33	56.00	71.11
Mean:	68.00	74.00	67.56	73.33	64.00	65.67

Table 2
Classification performance of the SA-ANN; the subscripts denote number of samples.

s/n	SA_{50}	SA_{100}	SA_{150}	SA_{200}	SA_{250}	SA_{300}
1	78.57	79.31	77.27	61.02	66.22	70.79
2	78.57	79.31	75.00	72.88	56.76	50.56
3	50.00	79.31	75.00	72.88	66.22	69.66
4	78.57	82.76	75.00	72.88	55.41	64.05
5	78.57	48.28	75.00	61.02	66.22	70.79
6	78.57	79.31	75.00	72.88	66.22	70.79
7	85.71	72.41	75.00	66.10	66.22	74.16
8	78.57	79.31	61.36	72.88	64.87	70.79
9	78.57	79.31	75.00	72.88	66.22	70.79
10	78.57	79.31	72.72	72.88	50.00	70.79
Mean:	76.43	75.86	73.64	69.83	62.44	68.32

Table 3
Classification performance of the LSTM-ANN; the subscripts denote number of samples.

s/n	$LSTM_{50}$	$LSTM_{100}$	$LSTM_{150}$	$LSTM_{200}$	$LSTM_{250}$	$LSTM_{300}$
1	80.00	80.00	24.44	26.67	56.00	71.11
2	00.00	00.00	00.00	73.33	73.33	71.11
3	73.33	80.00	00.00	26.67	66.67	06.67
4	80.00	80.00	75.56	56.67	56.67	71.11
5	20.00	46.67	75.56	25.00	00.00	71.11
6	20.00	00.00	33.33	10.00	52.00	00.00
7	80.00	20.00	62.22	66.67	18.67	22.22
8	20.00	80.00	00.00	45.00	52.00	71.11
9	20.00	20.00	31.11	00.00	34.67	00.00
10	80.00	20.00	75.56	11.67	50.67	28.89
Mean:	47.33	42.67	37.78	34.17	46.07	41.33

Table 4
Compared maximum and minimum classification accuracies.

Neural scheme	CA_{max}	CA_{min}
HTM-CLA	80.00	20.00
SA-ANN	**85.71**	48.28
LSTM-ANN	80.00	00.00

50 as against 5iterations used in the HTM-CLA and SA-ANN techniques; this was done to make the LSTM-ANN more competitive but at the price of increased computational time.

The classification accuracy results given in Tables 1–3 are for 10 trial runs and at different sample sizes from 50 to 300; the samples are incremented row-wise at a width of 50 samples. From the tables (Tables 1–3) the net mean average classification accuracies of the HTM-CLA, SA-ANN and the LSTM-ANN are 68.76%, 71.09% and 41.56% respectively.

The results comparing HTM-CLA with SA-ANN and LSTM-ANN for the German CCF data in terms of the maximum possible accuracies obtainable are also as presented in Table 4. Also, the worst case (minimum) possible accuracies are reported.

In Fig. 8, the average classification accuracies of each considered number of training examples are also presented.

5.3. Results using the Australian credit card benchmark

The comparative classification performance obtained using the Australian dataset is as presented in Table 5. Also, the maximum and minimum possible accuracies obtainable are as presented in Table 6. The entire (full) dataset are used for the analysis.

5.4. Discussions

The experiments of using a more biologically inspired neural technique called the Hierarchical Temporal Memory (HTM) for classifying credit card fraud have shown some promising results. As presented in the aforementioned sub-sections (Sections 5.2 and 5.3) the competitiveness of the proposed HTM technique as compared with the SA-ANN and a deep recurrent LSTM-ANN have shown that using a more biological neural technique is as effective as the conventional. Though, the performances of the HTM and SA-ANN techniques may differ, they are generally or roughly equivalent on the average.

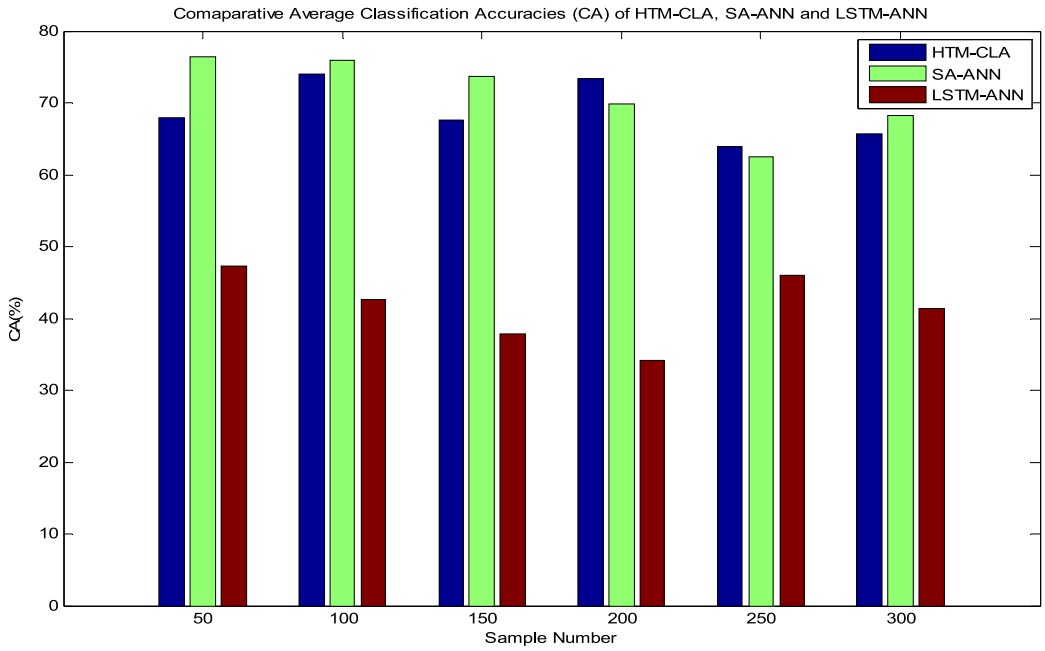

Fig. 8. Comparative accuracies of the HTM-CLA, SA-ANN and LSTM-ANN techniques.

Table 5
Classification performance of the HTM-CLA, SA-ANN and LSTM-ANN techniques.

s/n	HTM-CLA	SA-ANN	LSTM-ANN
1	56.04	51.94	38.65
2	54.11	56.31	43.96
3	46.38	56.31	39.13
4	48.31	56.80	34.30
5	48.79	56.31	40.10
6	50.72	57.28	40.10
7	44.93	56.31	19.81
8	43.96	56.31	37.68
9	51.69	56.31	09.18
10	45.89	56.31	00.00
Mean:	49.08	56.02	30.29

Table 6
Compared maximum and minimum classification accuracies.

Neural scheme	CA_{max}	CA_{min}
HTM-CLA	56.04	43.96
SA-ANN	**57.28**	51.94
LSTM-ANN	43.96	00.00

5.4.1. Studies on the german credit card fraud data

The results from Table 1 show that increasing sample size may lead to improvement in the classification performance of HTM-like algorithms; this however depends on the pattern of sequences considered as is can be seen in Table 1, there are some exceptions when progressing from a sample size of 100 to 150 and from 200 to 250. The average classification performance ratio (ACPR) of HTM-CLA using 300 samples over that using 250 samples is about 1.03. Though this ratio is small, it is expected that higher samples will lead to much higher ACPRs.

Comparing results in Tables 1 and 2, the HTM-CLA gave better average classification accuracy over the SA-ANN when the sample sizes are 200 and 250 samples. However, for the rest of the classification accuracies, the SA-ANN fared better than that of the HTM-CLA technique; the reason for the SA-ANN improved performance may be attributed to its desirable random annealing initialization step and an inherent pattern shuffling operation. Also, both HTM-CLA and SA-ANN techniques fared much better than the LSTM-ANN; this is clearly evident in Fig. 7.

The SA-ANN gave the highest accuracy with a value of 85.71% (see value in bold - Table 4); however, this is dependent on the nature of the considered sample size as is shown in the results of Tables 1–3.

In general, the net ACPR of HTM-CLA over that of the SA-ANN and LSTM-ANN techniques are 0.97:1 and 1.66:1 which is approximately 1:1 and 2:1 respectively. Thus, the HTM-CLA is equivalent to the SA-ANN and two-times better than the LSTM-ANN techniques.

The interesting result interpretations indeed show that HTM-CLA is a promising technique based on the considered dataset and system parameter specifications. This may be attributed to its high representational capacity when compared to the SA-ANN or LSTM-ANN approaches. It also validates the unique importance of SDRs as the main representational unit in HTM-CLA.

5.4.2. Studies on the Australian credit card fraud data

The comparative performance of the HTM-CLA, SA-ANN and LSTM-ANN techniques are as shown in Table 5. From the result, the ACPR of HTM-CLA over that of the SA-ANN and LSTM-ANN techniques are 0.88:1 and 1.62:1 which is approximately 1:1 and 2:1 respectively. Thus, just as in the results using the German CCF data, the HTM-CLA is equivalent to the SA-ANN and two-times better than the LSTM-ANN techniques.

5.5. Practical implications and recommendations

This research has presented a more biologically inspired neural technique called the Hierarchical Temporal Memory (HTM) for classifying credit card fraud. This approach is well suited for real time analysis and can be integrated in real world financial systems such as found in banks and financial organizations that require an online (continual detection) of fraudulent patterns.

In this respect, using the ideas in this research, the financial records of a given customer can be encoded in the form of sparse distributed representations (SDRs) by an especially designed scalar encoder and the resulting data streams continually tracked to identify anomalies in the use of financial instrument such as credit cards. The bank administrators and/or information security personnel can then leverage on this information to conduct appropriate remedial actions.

In particular, it is also possible that the entire system will be automated in such as to interactively disable credit card transaction systems and assure the integrity of the entire financial system.

Table A.1
Attributes of simulation data (German credit dataset).

Attribute number	Attribute name	Attribute description
1.	Status of existing checking account	Qualitative
2.	Duration in month	Numerical
3.	Credit history	Qualitative
4.	Purpose of credit transaction	Qualitative
5.	Credit amount	Numerical
6.	Saving account/bonds	Qualitative
7.	Present employment	Qualitative
8.	Installment rate in percentage of income	Numerical
9.	Personal status & sex	Qualitative
10.	Other debtors/guarantors	Qualitative
11.	Period of present residence	Numerical
12.	Property	Qualitative
13.	Age in years	Numerical
14.	Other installment plans	Qualitative
15.	Housing	Qualitative
16.	Number of existing credits at the bank	Numerical
17.	Job status	Qualitative
18.	Number of people liable to provide maintenance for	Numerical
19.	Telephone	Qualitative
20.	Foreign worker (Yes or No)	Qualitative
21.	Class label	Numerical (1 — valid; 2 — invalid)

Table A.2
Attributes of simulation data (Australian credit dataset).

Attribute number	Attribute name	Attribute description
1.	Undisclosed	Categorical
2.	Undisclosed	Continuous
3.	Undisclosed	Continuous
4.	Undisclosed	Categorical
5.	Undisclosed	Categorical
6.	Undisclosed	Categorical
7.	Undisclosed	Continuous
8.	Undisclosed	Categorical
9.	Undisclosed	Categorical
10.	Undisclosed	Continuous
11.	Undisclosed	Categorical
12.	Undisclosed	Categorical
13.	Undisclosed	Continuous
14.	Undisclosed	Continuous
15.	Undisclosed	Numerical (1 — valid; 2 — invalid)

5.6. Encoding technique

The encoder is a key feature that must be well adapted to the raw input data to form the equivalent binary SDRs needed in HTM-like anomaly processing systems (Hawkins et al., 2016; Purdy, 2016). In this research we have applied a new but yet simple and powerful approach to encoding SDR sequences based on reference equivalency matching (REM). This approach was experimentally found to generate a unique set of input SDRs for each unique feature pattern.

Since the dataset is composed of a number of string sequences (textual representation of features), we transform the strings into numeric integers based on the standard ASCII set. This is then followed by data normalization to a range between 0 and 1 to form a potential set of input synapses.

The normalization operation to form the potential input synapses is simply defined as:

$$A_{synapse} = \frac{A_{store}}{n_f} \tag{4}$$

where,
$n_{f=}$ normalization factor
A_{store} = numeric representation of the dataset input strings
We use the reference-equivalency matching (REM) algorithm (see Listing 1) to form the actual input SDRs for each feature sequence in a number of observations where the features are depicted as columnar entities and the observations as row-wise patterns. In this algorithm, the elements in the first (or last) row of the normalized input feature data are used as reference and for each observed (transformed) sequence the reference is iteratively compared with the subsequent rows and the corresponding logical 0 or 1 states stored in an SDR holding variable.

6. Conclusion and future work directions

In this research paper, an emerging and living neurocomputing technology technique for solving the credit card fraud (CCF) anomaly detection problem — the Hierarchical Temporal Memory based on Cortical Learning Algorithms (HTM-CLA) is presented. The HTM-CLA has been adapted to sequential time series predictive classification of sample German CCF dataset.

The results using this emerging technology on the case study data show that there may be a potential improvement in the computed ACPR if higher sample data sizes are used. Thus, future research should further investigate the incremental effect of sample size on HTM-CLA performance.

Also, this work has compared the performance of HTM-CLA with a conventional neural technique trained with the Simulated Annealing (SA) approach (SA-ANN). The HTM-CLA is competitive with the SA-ANN technique with an ACPR of approximately 1:1. However, the maximum accuracy can be obtained from the SA-ANN technique. Thus, the HTM-CLA technique may not entirely be a better technique when compared with other fundamental neural network schemes.

In general, all considered techniques have potential in CCF detection if the architecture given best accuracy can be employed in real life situation. However, the HTM-CLA offers more realistic features in terms of systems design and robustness when considering the deployment in real world CCF detection schemes.

In particular, future work may be directed towards optimizing the HTM-CLA parameters to improve its prediction error response and studying the computational run times of HTM-CLA prediction algorithms in comparison with other similar neurobiological continual learning neural techniques. This will ensure that the future HTM-CLA variants are not only robust in terms of representational capacity, but they are faster and more accurate with optimal parameterization. Hybridization schemes are also recommended where the HTM-CLA with a conventional ANN such as the SA-ANN are combined in solving sequence learning streaming data problems.

Another challenging area is reducing the computational burden on the HTM-CLA algorithm — this border specifically on the effect of the data attribute size on the run-time performance of the Spatial-Pooling (SP) stage of the algorithm. Attempts at attribute reduction where only the most essential attributes are considered for prediction are also currently under investigation by the authors.

Listing 1: Reference- equivalency matching algorithm

Initialize sequence size, s_z, I, as sequence iterator index, J, as reference index, A_{hold} as SDR holding variable

for all $i \in$ I, $j \in$ J, I, J $\in s_z$

Set $j = 1$

Compute $A_{hold(i)} = (A_{store(j)} == A_{store(i)})$

end for

Table B.1
Key default HTM-CLA system parameters.

Parameter number	Parameter name	Used parameter value (German CCF data)	Used parameter value (Australian CCF data)
1.	Initial permanence	0.81	0.51
2.	Connected permanence	0.50	0.50
3.	Permanence increment	0.10	0.10
4.	Permanence decrement	0.01	0.01
5.	Number of columns	256	256
6.	Stimulus threshold	15	2
7.	Boost factor	100	100
8.	Iterations	5	5

Table B.2
Key default SA-ANN system parameters.

Parameter number	Parameter name	Used parameter value (German and Australian CCF data)
1.	Hidden neuron size	1 to 10
2.	Number of function evaluations	1,000,000
3.	Starting temperature	−1000 to −500
4.	Number of sample iterations	5

Table B.3
Key default LSTM-ANN system parameters.

Parameter number	Parameter name	Used parameter value (German CCF data)	Used parameter value (Australian CCF data)
1.	Layer size	4	4
2.	Batch size	35	450
3.	Iterations	50	50

CRediT authorship contribution statement

E.N. Osegi: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Visualization, Investigation, Supervision, Software, Validation. **E.F. Jumbo:** Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This study received no funding from any source or agency. The authors wish to thank the reviewer team and the Editors for their constructive comments which greatly helped in the development of this research paper.

Appendix A

See Tables A.1 and A.2.

Table B.4
LSTM-ANN system learning rate parameters.

Parameter number	Sample size	Learning rate parameter value
1.	50	0.000055
2.	100	0.000035
3.	150	0.000020
4.	200	0.000015
5.	250	0.000010
6.	300	0.000005

Appendix B

See Tables B.1–B.4.

References

Agrawal, P., & Franklin, S. (2014). Multi-layer cortical learning algorithms. In *2014 IEEE symposium on computational intelligence, cognitive algorithms, mind, and brain* (pp. 141–147). IEEE.

Ahmad, S., & Hawkins, J. (2015). Properties of sparse distributed representations and their application to hierarchical temporal memory. arXiv preprint arXiv:1503.07469.

Anireh, V. I. E., & Osegi, E. N. (2017). A modified activation function with improved run-times for neural networks. *Advances in Multidisciplinary & Scientific Research Journal*, *3*(2), 33–44.

Budhram, T. (2012). Lost, stolen or skimmed: Overcoming credit card fraud in South Africa. *South African Crime Quarterly*, *40*, 31–37.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 1–58.

Cui, Y., Ahmad, S., & Hawkins, J. (2017). The HTM spatial pooler—A neocortical algorithm for online sparse distributed coding. *Frontiers in Computational Neuroscience*, *11*(111).

Falaki, S. O., Alese, B. K., & Ismaila, W. O. (2010). An update research on credit card on-line transactions. *International Journal of Economic Development Research and Investment*, *1*(2 & 3).

Fayek, H. M. (2019). *Continual deep learning via progressive learning* (PhD Dissertation), RMIT University.

Galetzka, M. (2014). *Intelligent predictions: An empirical study of the cortical learning algorithm*. (Msc Dissertation), University of Applied Sciences Mannheim.

García-Pedrajas, N., Ortiz-Boyer, D., & Hervás-Martínez, C. (2006). An alternative approach for neural network evolution with a genetic algorithm: Crossover by combinatorial optimization. *Neural Networks*, *19*(4), 514–528.

Hawkins, J., Ahmad, S., & Dubinsky, D. (2011). *Hierarchical temporal memory including htm cortical learning algorithms*: *Techical report*, Palto Alto: Numenta, Inc, http://www.numenta.com/htmoverview/education/HTM_CorticalLearningAlgorithms.pdf.

Hawkins, J., Ahmad, S., & Dubinsky, D. (2014). *The science of anomaly detection*: *Online technical report*, (pp. 1–18). Redwood City, CA: Numenta, Inc, Available: http://numenta.com/learn/science-of-anomaly-detection.html.

Hawkins, J., Ahmad, S., Purdy, S., & Lavin, A. (2016). *Biological and machine intelligence (initial online release 0.4)*. Numenta Inc.

Hawkins, J., & Blakeslee, S. (2004). *On intelligence* (1st ed.). NY: Times Books.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hofmann, H. (1996). *German credit data. uci repository of machine learning databases*. Irvine, CA: Department of Information and Computer Science, University of California, https://archive.ics.uci.edu/ml/datasets/statlog?(german?credit?data.

Kamaruddin, S., & Ravi, V. (2016). Credit card fraud detection using big data analytics: Use of PSOAANN based one-class classification. In *Proceedings of the international conference on informatics and analytics* (pp. 1–8).

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

McCall, R., & Franklin, S. (2013). Cortical learning algorithms with predictive coding for a systems-level cognitive architecture. In *Second annual conference on advances in cognitive systems poster collection* (pp. 149–66).

Osegi, E. N. (2021). Using the hierarchical temporal memory spatial pooler for short-term forecasting of electrical load time series. *Applied Computing and Informatics*, *17*(2), 264–278. http://dx.doi.org/10.1016/j.aci.2018.09.002.

Osegi, E. N., & Anireh, V. I. (2020). AMI: An auditory machine intelligence algorithm for predicting sensory-like data. *Anatolian Journal of Computer Sciences*, *5*(2), 71–89.

Patidar, R., & Sharma, L. (2011). Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering*, *1*(32–38).

Purdy, S. (2016). Encoding data for HTM systems. arXiv preprint arXiv:1602.05925.

Quinlan, J. R. (1987). *Australian credit dataset. UCI repository of machine learning databases*. Irvine, CA: Department of Information and Computer Science, University of California, http://archive.ics.uci.edu/ml/datasets/statlog?(australian?credit?approval.

Quinlan, J. R. (1992). C4. 5: Programs for machine learning, In *The Morgan Kaufmann series in machine learning*, (1st ed.).

Sexton, R. S., Dorsey, R. E., & Johnson, J. D. (1999). Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated algorithm and simulated annealing. *European Journal of Operational Research*, *114*(3), 589–601.

Shahookar, K., & Mazumder, P. (1991). VLSI cell placement techniques. *ACM Computing Surveys*, *23*(2), 143–220.

Singh, S., Karnwal, A., Prasad, N., D'Souza, R., & Shenoy, A. (2006). Fraud detection using neural network. In *Proceedings, national conference CISCON-06*. Manipal, India.

Wang, C., Wang, Y., Ye, Z., Yan, L., Cai, W., & Pan, S. (2018). Credit card fraud detection based on whale algorithm optimized BP neural network. In *2018 13th international conference on computer science & education* (pp. 1–4). IEEE.

Zou, J., Zhang, J., & Jiang, P. (2019). Credit card fraud detection using autoencoder neural network. arXiv preprint arXiv:1908.11553.