ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/352973630

A Perceptron Based Neural Network Data Analytics Architecture for the Detection of Fraud in Credit Card Transactions in Financial Legacy Systems

Article *in* WSEAS Transactions on Systems and Control · July 2021

DOI: 10.37394/23203.2021.16.31

CITATIONS
0

READS
97

2 authors, including:

Raul Valverde
John Molson School of Business, Concordia University & Université du Québec en Outaouais
113 PUBLICATIONS **506** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Cybercompetencies View project

All content following this page was uploaded by Raul Valverde on 05 July 2021.

The user has requested enhancement of the downloaded file.

A perceptron based neural network data analytics architecture for the detection of fraud in credit card transactions in financial legacy systems

QUINTIN-JOHN SMITH,
Temenos, USA 300 Primera Boulevard
Suite 444
Lake Mary, Florida, 32746 USA

RAUL VALVERDE
John Molson School of Business
Concordia University
1450 Guy St
Montreal Quebec H3H 0A1 CANADA

Abstract: - Credit card fraud, a significant and growing problem in commerce that costs the global economy billions of dollars each year, has kept up with technological advancements as criminals devise new and innovative methods to defraud account holders, merchants, and financial institutions. While traditional fraudulent methods involved card cloning, skimming, and counterfeiting during transactional processes, the rapid adoption and evolution of Internet technologies aimed at facilitating trade has given rise to new digitally initiated illegitimate transactions, with online credit card fraud beginning to outpace physical world transactions. According to the literature, the financial industry has used statistical methods and Artificial Intelligence (AI) to keep up with fraudulent card patterns, but there appears to be little effort to provide neural network architectures with proven results that can be adapted to financial legacy systems. The paper examines the feasibility and practicality of implementing a proof-of-concept Perceptron-based Artificial Neural Network (ANN) architecture that can be directly plugged into a legacy paradigm financial system platform that has been trained on specific fraudulent patterns. When using a credit checking subscription service, such a system could act as a backup.

Key-Words:: Neural networks, fraud detection, financial legacy systems

Received: June 5, 2021. Revised: June 24, 2021. Accepted: June 28, 2021. Published: July 2, 2021.

1 Introduction

Data mining plays an important role for both the technology and businesses. It can present complex information into something useful, easy to understand and reflect the most up-to-date status in real-time. Furthermore, it can help the organizations to achieve its goals, improve its business and enhance products and services. One important area is the credit card fraud detection and analysis enabled by data mining. This paper applies business analytics procedures focused on credit card fraud detection commonly found inside the retail industry by utilizing a business retailer legacy system situated in South Africa as the fundamental contextual investigation. The examination centers in the formation of a data mining, visual representation and an adaptable and viable programming solution, consisting of a few loosely coupled separate modular components that could be integrated in legacy financial systems for credit card fraud detection. By using an object orientated methodology, the paper presents a completely adaptable and configurable Artificial Neural Network (ANN) suite [1], ready to plug straightforwardly into a legacy paradigm [2]. The outcome would be a financial system that can help with the identification of fraudulent business credit card transactions.

Credit card companies have executed different techniques to lessen fraud in business transactions. This incorporates fuzzy logic offered as membership systems, including chip-n-pin [3], contactless [4] and 3D Secure [5], challenge-and-response procedures, presently consolidating mobile and credit card purchases through GPS (Global Positioning System) that match purchase location against user’s data [6].

Yet, Zetter [7], contends that fraud done face to face with a card have dropped since criminals can't utilize fake cards with stolen information embossed on them any longer and cautions that fraud including card not present transactions has altogether increased. Shankland [6], gives that any significant expense, low-return estimates used to recognize fraud won't be implemented if the expense surpasses the likely gains from fraud decrease and means we are in a dilemma where solutions should be successful at identifying fraud yet cost effective, as degree of profitability versus accuracy [8] is a significant metric that must be considered.

Patidar and Sharma [9], depicts that while data mining procedures are utilized to discover relationship in datasets [10], data collected, and classification requires extra off-line processing and data analysis to comprehend and perceive what has been clustered. Also, knowledge discovery made through data analysis, requires a comprehension of data partitioning and correlation problems [11] prior to creating rule-based systems based on observations and patterns. Zhang and Bivens [12] performed scientific comparisons between Bayesian Networks (BN) and Artificial Neural Networks (ANN) explicitly in performance, inferring that BN are less sensitive to small datasets with incomplete information, and have better performance for conditions that change quickly. Nonetheless, ANN albeit less exact and subject to training methods can achieve quicker model assessment times, appropriate for conditions that request require high volume rea-time predictive analytics [12].

There are many comparative studies of classification algorithms in the literature for similar fields such as credit card fraud and Intrusion Detection systems, and so on. A well known comparative study of the classification algorithms was carried out in the STATLOG project (Michie et al, 1994) involving 22 real world data sets and about 23 different algorithms. One of the main observations in the study was that many of the classification algorithms compared performed best over a collection of datasets but there was no unique best classifier across all datasets. And the performance of each algorithm depended on the characteristics of the dataset.
Here

Maes et al (2000) compared artificial neural networks and Bayesian networks in their ability to detect credit card fraud. Maes et al observed that Bayesian networks performed better than neural networks in detecting credit card fraud.

Shavlik, Mooney and Towell (1991), have compared the performance of the ID3 type of Decision Tree algorithm with the back propagation and perceptron neural networks algorithms on five real world data sets in terms of classification accuracy. They found that the back propagation neural network algorithm performed better than the other two algorithms even though it takes a longer time to train. Dietterich, Hild and Bakri (1995), have also compared the performance of the back propagation neural network algorithm with that of the ID3 type of Decision Tree algorithm on the task of mapping English text to phonemes and stresses. They have observed that the back propagation neural network algorithm consistently out performed the ID3 Decision tree algorithm by several percentage points.

Lim, Loh and Shin (2000), have compared 22 decision trees, 9 statistical and 2 neural networks algorithms on 32 data sets using classification accuracy as a measure. One of the statistical algorithms called poly class was the best of all algorithms evaluated. Other top ranked algorithms included linear regression, logical discriminant analysis and Quest with linear splits Decision Tree algorithm.
End
Ostensibly, for any credit card transaction fraud recognition to be viable, any implemented system should assess and predict on known data patterns in real time without disrupting the execution of the credit card processing system [9]. Also, ANN as indicated by Zhang and Bivens [12], performs better with large data volumes containing skewed data distributions [13], and as fraudulent business transactions are a little part of this volume [10] for this research, we trust ANN to be appropriate. As indicated by Joshi et al. [14], although the main application of ANN is in machine learning, pattern recognition is also an important application of this technology and for this reason suitable for our research.

ANN have been utilized recently for fraud detection in the financial business, Gómez et. al. [15] executed an ANN based framework for fraud detection in the banking industry by focusing on unevenness, data processing and cost metric assessment. Wang,et. al. [16] proposed a fraud detection system based on an ANN neural organization targeting tackling the issues of slow convergence rate, easy to fall into local optimum, network defects and poor system stability. Zhang et. al. [17] proposed a fraud detection model based on ANNs which increases the fraud detection performance comparing with existing condensed nearest neighbor data mining-based algorithms. Jurgovsky et. al. [18] proposed the fraud detection problem as a sequence classification tasks and employs Long Short-Term Memory networks to incorporate transaction sequences and integrates feature aggregation strategies.

Data mining suggests the extensive use of data, to drive decisions and actions [19]. Data mining has a history of successful application in fraud detection, for instance, Massa and Valverde [20] proposed a new data mining software-based system for fraud detection based of anomaly detection data analysis techniques with the purpose to predict computer intrusion attacks in web applications. Also, Kraus and Valverde [21] were able to successfully implement a data visualization and data mining system based on database store procedures in SQL and business intelligence visualization tools for the detection of fraud in procurement transactions in supply chain operations management.

A legacy system is an older software application that uses an obsolete hardware and software platform which is hard and expensive to maintain and integrate with modern systems [22]. Many challenges exist with legacy systems mainly because their lack ability to adapt to modern trends towards data mining, enterprise systems and supply chain systems [23]. Picton [24] and Gallant [25] describe Artificial Neural Networks (ANN) as “complex distributed parallel processing systems”, the implementation of an ANN can be quite challenging as it requires a complex arrangement of components that need to interact with each other.

The objective of this paper is to create an IT Artefact with different levels of abstraction that has the intention of reducing the inherent complexity normally associated with designing, training, validating and implementing ANN systems by providing a simple intuitive interface. This IT Artefact is intended also to integrate with existing financial legacy systems with minimal disruption or impact to currently active and implemented processes, workflow, and business logic. The intention of the research is to contribute to literature by providing a software ANN solution that can be used to extend the functionality of existing legacy financial systems with the purpose of providing fraud detection services.

Similarly, while the IT Artefact developed for this paper can generate an array of single level perceptrons (SLPs) and multi-level perceptions (MLPs), it may cause issues such as creating networks with varying connectional patterns [26] and varying topologies based on configurable parameters. According to [27], [28], and [29], credit card fraud and detection can be classified as a "non-linear data classification and regression problem." Because it has more research values for investigations, this paper will show the design, conceptualization, implementation, and testing of an MLP ANN that can be easily integrated in a financial legacy system, which is comprised of an input and output layer using a single hidden layer supplemented with bias neurons to identify and classify fraudulent credit card transactions.

The paper presents a data mining and visualization artefact developed and modelled on the perceptron algorithm [1] for a financial legacy system, using on-line supervised training by using the feed forward backpropagation algorithm (Bentley et al., 2000) [30][31], and momentum [32], that can learn from and classify transactional credit card data patterns provided and detect fraudulent transactions.

Another objective of this paper is to use techniques for pattern recognition and credit card data classification commonly found in the financial and retail industries to identify fraudulent credit card transactions using a case study of a large commercial and public retailer and by implementing a flexible and configurable Artificial Neural Network (ANN) software component that can
Materials and procedures
A large commercial and public retailer extracted credit card transactional data from their production database, yielding 13,9 million transactions totaling R8,9 billion over a two-year period, as shown in Figure 1. (2014-2016).

After analyzing data totals provided by large commercial and public retailers in Figure 1, it was discovered that fraudulent credit card transactions were at a percentage of 0.0977 percent, which is slightly higher than The UK Cards Association (2013), which states that credit card fraud in the United Kingdom has increased to 7,4 pence per hundred pounds (0.074 percent). However, as described by [34, 35], the distribution of fraudulent transactions within total transactional volume is skewed, with the extracted dataset showing a comparative total of 0.0501 percent fraudulent transactions. While reported figures from large commercial and public retailers may be similar with those provided by The UK Cards Association (2013), it should be noted that our study operates in a different geographical location, and the data provided for this paper is subject to transactional patterns unique to the industry in which they operate. Typically, for most real world domains the training dataset would closely represent the density and distribution of the production dataset.

Therefore, to create such a representative set of the data, random sampling was used. Furthermore, random sampling provides an unbiased view of the data. However, random sampling may not work well especially when modeling rare occurrences within the data.

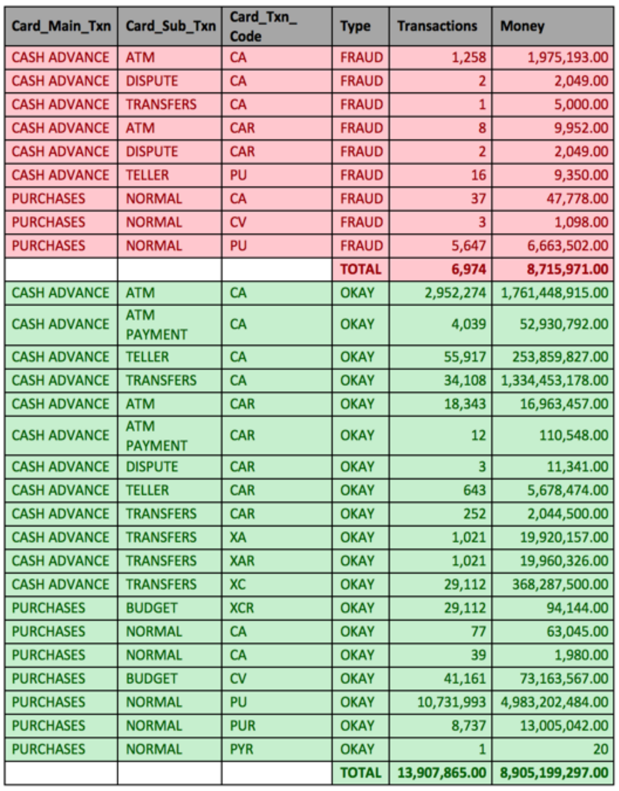
Due to relatively low levels of fraudulent transactions in comparison to total credit card volumes, the detection and classification methods using predictive systems capable of handling skewed distributions [34][35], as well as noise and classification overlaps [36], that are easily trainable, testable, verifiable, and upgradeable, are required. This study investigated the feasibility and accuracy of using a custom software application based on object-oriented design and by using the Java language and Perceptron ANN algorithm [1], modelled, MLP, that was specifically developed as an IT Artefact by using standard feed-forward, supervised on-line training with momentum [32] as a learning algorithm.

The software solution was designed and implemented as a modular, loosely coupled and interchangeable component-based ANN suite, applying several commonly used design patterns as described by [38], and trained to be able to correctly classify and identify fraudulent transactions within datasets provided by a case study located in South Africa, with data spanning over a two-year period (2014-2016). The software module custom developed was integrated into financial legacy systems to visualize and detect fraud using the trained ANN. All collected the data for this study has been independently audited and confirmed by the firm Mazars (leading international audit, tax and advisory firm) in South Africa.

The datasets provided was divided into three sections:

1. Training data that consists of data and target values that are presented to the system for learning purposes.
2. Generalization data that has not previously been seen by the network but whose target values are known to the evaluators.
3. Evaluation data that will be used to simulate a production environment, consisting of data that has not been seen by the system or used during training and generalization but known to the evaluators as target values.

2 Problem Formulation



Card_Main_Txn	Card_Sub_Txn	Card_Txn_ Code	Type	Transactions	Money
CASH ADVANCE	ATM	CA	FRAUD	1,258	1,975,193.00
CASH ADVANCE	DISPUTE	CA	FRAUD	2	2,049.00
CASH ADVANCE	TRANSFERS	CA	FRAUD	1	5,000.00
CASH ADVANCE	ATM	CAR	FRAUD	8	9,952.00
CASH ADVANCE	DISPUTE	CAR	FRAUD	2	2,049.00
CASH ADVANCE	TELLER	PU	FRAUD	16	9,350.00
PURCHASES	NORMAL	CA	FRAUD	37	47,778.00
PURCHASES	NORMAL	CV	FRAUD	3	1,098.00
PURCHASES	NORMAL	PU	FRAUD	5,647	6,663,502.00
			TOTAL	**6,974**	**8,715,971.00**
CASH ADVANCE	ATM	CA	OKAY	2,952,274	1,761,448,915.00
CASH ADVANCE	ATM PAYMENT	CA	OKAY	4,039	52,930,792.00
CASH ADVANCE	TELLER	CA	OKAY	55,917	253,859,827.00
CASH ADVANCE	TRANSFERS	CA	OKAY	34,108	1,334,453,178.00
CASH ADVANCE	ATM	CAR	OKAY	18,343	16,963,457.00
CASH ADVANCE	ATM PAYMENT	CAR	OKAY	12	110,548.00
CASH ADVANCE	DISPUTE	CAR	OKAY	3	11,341.00
CASH ADVANCE	TELLER	CAR	OKAY	643	5,678,474.00
CASH ADVANCE	TRANSFERS	CAR	OKAY	252	2,044,500.00
CASH ADVANCE	TRANSFERS	XA	OKAY	1,021	19,920,157.00
CASH ADVANCE	TRANSFERS	XAR	OKAY	1,021	19,960,326.00
CASH ADVANCE	TRANSFERS	XC	OKAY	29,112	368,287,500.00
PURCHASES	BUDGET	XCR	OKAY	29,112	94,144.00
PURCHASES	NORMAL	CA	OKAY	77	63,045.00
PURCHASES	NORMAL	CA	OKAY	39	1,980.00
PURCHASES	BUDGET	CV	OKAY	41,161	73,163,567.00
PURCHASES	NORMAL	PU	OKAY	10,731,993	4,983,202,484.00
PURCHASES	NORMAL	PUR	OKAY	8,737	13,005,042.00
PURCHASES	NORMAL	PYR	OKAY	1	20
			TOTAL	**13,907,865.00**	**8,905,199,297.00**

Figure 1 – Data extract summary, large commercial retailer showing breakdown and totals per transaction type.

Because the case study's data distributions show a skewed gradient of 0.0501 percent, where 99,9499 percent of transactions are legitimate, resulting in difficult training scenarios and slow convergence rates (Bentley et al., 2000), the number of legitimate transactions was reduced, providing a higher percentage of fraud ratio of 13,94%. The data was divided into training samples with less skewed data (Stolfo et al.,1997) and a fraud / non-fraud dataset ratio of (6974 (fraud) /50000 (legitimate)).

Naturally, any implemented system requires several metrics, and this study focused on metrics capable of measuring not only classification accuracy but also algorithm cost functions during convergence processes via back-propagation [30][31], with momentum [32].
According to Twomey and Smith [39], forward feed, backpropagation neural networks are one of most used neural networks learning algorithm that has been successfully applied to a wide range of industries and sectors within commerce, able to solve pattern recognition and classification problems with varying degrees of success.

For example, Ghosh and Reilly [40] conducted research using a forward feed, backpropagation ANN to detect credit card fraud using large datasets provided by a card issuer (Mellon Bank). They demonstrated that such system could successfully classify lost, stolen, counterfeit, and mail-order fraud over a two-month period. White [41], on the other hand, adds that ANN systems must have some associated metric or validation process to ensure data quality, network design complexity, and generalization within the network to work as expected, producing and returning output vectors with some level of accuracy.

Duro and Reyes [42] suggest that three of the most commonly used metrics for the validation and verification of Neural Networks are the Mean Absolute Error (MAE) [43], Root Mean Squared Error (RMSE) [44], and a percentage classification [45], where the network's "goodness" is measured by comparing network output vectors against known and expected target values and in a similar manner.

According to Duro and Reyes [42], while there is “no well-formulated” or theoretical methodology for ANN model valuation, the proposed metrics can encompass all variations and aspects of neural network design. According to Tino, Cernansky, and Benuskova [46], the standard practice is to base model validation on specific network performance measurements (NPM) based on data that was not used in model construction [47]. Similarly, this study used a technique in which both validation and test data were not included in the actual training data sets used to determine the network's ability to generalize on previously unseen results. The study also employs a mean percentage estimator (MPE) computed and summed [48], which is used to determine the level of accuracy and generalization of the neural network using a binary output [42] approach. Defined and mapped output neurons are expected to produce results in the range of zero to one, with one representing a positive result (neuron has fired) and zero representing a negative result (neuron has not fired). When vectors returned by output neurons have a value greater than 0.5, it is assumed that the output was one, and when the value is less than 0.5, a zero value is assumed [41].

White [41] suggests that while MPE estimators [45], MAE [43], and RMSE [44] are good approximations and indicators of network performance, it is also a good practice to show cost functions and learning curve estimators.

To improve generalization of results for unseen data, training tolerance levels and margins may need to be adjusted in many cases. For this purpose, this study provided an adjustable momentum [49], learning [50], RMSE, and epoch rates that could be used to refine results and improve generalization, resulting in a better distributed Gaussian function approximation.

Tetko [51] warns that Perceptron-based ANN models are prone to over-fitting, which is defined as a problem that occurs when “a network describes noise rather than the underlying data relationships”. Leinweber [52] also suggests that over-fitting may occur in network models with excessive topology complexity or in network models with too many defined parameters (neurons / nodes) against inherently less complicated datasets, resulting in poor predictive generalization qualities or low performance results.

There are limitations for the chosen ANN model for this study and for this purpose, different precautionary measures can be included in the IT Artefact as configurable parameters capable of applying weight decay [53], dropout or pruning [54], and data shuffling [55]. This study, however, did not use early stopping [56] during network training because this could be described as a manual procedure and verification process requiring some knowledge and competency in neural network training paradigms, and the proposed implementation IT Artefact was designed for the layman or non-neural network expert users.

2 Neural network framework and design

To create a fully scalable and maintainable system, it is proposed that the system should be built with separate and interchangeable components that can be swapped out without affecting the rest of the software, with the goal of integrating this system with a legacy financial system for data mining purposes. Taking this into consideration, it is proposed that a modular system be developed, consisting of several loosely coupled and integrated components, each housing and encapsulating their own business logic, which when combined will implement a fully functional neural network suite that can be upgraded at any time without affecting other components in the legacy financial system.

Keeping modularity, encapsulation of business logic, and system integration in mind, the study implemented three IT Artefacts which include the following:

1. A Neural Network: encapsulated with its own business logic that uses feed-forward to compute and return vector results based on input parameters passed, as well as a back-propagation function that uses the gradient descent chain-rule to parameterize and train the network within error margins defined by users. Although only one learning algorithm was implement, users can upload and use pre-configured and trained algorithms.
2. A Desktop Visualization: Application installed on a local machine that can be used to visualize the neural network to the user as well as allow users for network parameterization, configuration, training, and validation prior to deployment. It is also proposed that the Desktop application serve as a framework rather than encapsulating any business logic, and that it is composed of several independent applications loaded at run-time that offer integration into a single application that represents the Neural Network and configurable parameters.

3. A Neural Interface for Data Mining: This resides on the legacy financial banking application server and responds in real-time to banking system requests, computing output vectors based on input vectors and passing algorithms. It is also proposed that the neural network be able to upload and process various algorithms passed in real-time, making such an interface generic and extensible beyond the scope of this study that provides flexibility and more general application of the proposed solution.

In addition to the previously identified Factory Design Pattern, which is used to create neurons based on user and system passed parameters, it is suggested that the Neural Network Framework employs the Prototype Design Pattern described by [57], which loads and creates neuron objects stored in cache at run-time. According to Fowler [38], such an approach would not only significantly improve system performance but also would also reduce estimated resources and overhead requirements by not having to create multiple objects that are already available and accessible in memory. The following design patterns have been identified to aid functionality and future scalability in the event that any individual component of the application requires enhancements or upgrades.

i. Front Controller Pattern: According to Freeman et al. [57], a front controller design pattern is used to provide a centralized request handling mechanism that allows all requests to be handled by a single handler. The research uses this design pattern for the same purpose, allowing the desktop application to act as a centralized handler capable of controlling the proposed plugin components.

ii. The Singleton Design Pattern: According to Freeman et al. [57], singleton design patterns, which are often regarded as the simplest design patterns in use today and fall under creational patterns, are one of the best ways to create a single class object that can be accessed directly without the need to instantiate such an object first. The research focuses on plug-in components that will be directly controlled by the Desktop Visualization Application, often consisting of a single class with known exposed public methods to facilitate message flow. The singleton pattern is seen as the best fit because it ensures that proposed solution will deal only with a single instance of each plug-in object type.

iii. The Template Design Pattern: According to Freeman et al. (2004), abstract classes can be used to expose pre-defined methods or events that can be used to implement and invoke messages between objects by using a Template Design Pattern, which is classified under the behavioral category. Similarly, the research employs the Template Design Pattern to provide a common referenced framework of known and exposed methods and call back events for the creation of additional and ad-hoc components that can be plugged into the Desktop Visualization Application. According to Freeman-Benson [57], one of the benefits of using this design pattern is that objects do not need to know about each other at design time because calls are only invoked at run-time.

The design employs different software design patterns that are used to not only create a flexible, scalable, and easily maintainable, upgradeable system, but design patterns and their application also solve commonly occurring architectural problems in the industry and provide reliable and proven methods to implement best design practices [38].

Considering the points made by [38] and Freeman et al. [57], the use and implementation of proposed design patterns for this research would significantly aid in system design, rather than simply hammering a solution into a pattern for the sake of using one.

A Neural Network Framework: incorporates two different design patterns and is used to dynamically create neuron objects based on system and user parameters passed. The Neural Network Framework can model simple and complex neural network design structures made up of individual neuron objects, each modelled after the Perceptron neuron model [1] as shown in Figure 2, and containing several functions, algorithms, and data rules derived from a common base class called Neuron.

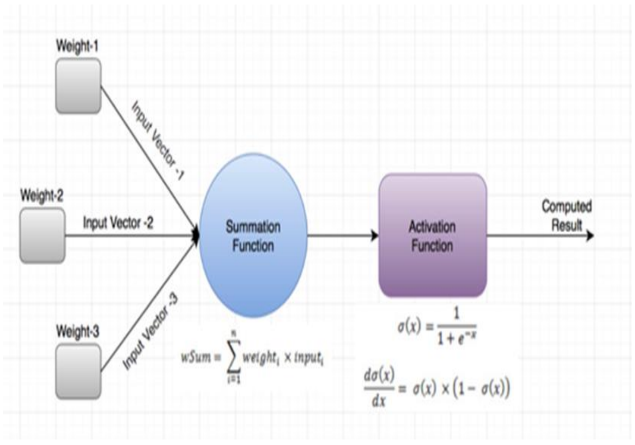

Weight-1
Input Vector -1
Weight-2
Input Vector -2
Weight-3
Input Vector -3
Summation Function
Activation Function
Computed Result
$wSum = \sum_{i=1}^{n} weight_i \times input_i$
$\sigma(x) = \frac{1}{1+e^{-x}}$
$\frac{d\sigma(x)}{dx} = \sigma(x) \times (1 - \sigma(x))$

Figure 2 The modern Perceptron Neuron Model, based on (Rosenblatt, 1958), showing input weights, summation and activation function (using a logarithmic sigmoidal).

The base component from which all neurons inherit their traits is simply defined as a Neuron Class within the Neural Network Framework, and although some of the methods are overridden in different Neuron types, depending on their functionality, the research uses the Factory Design Pattern [38] to create Neurons based on type (Input, n-Hidden, and Output), and such a design pattern would allow Figure 3 depicts a class diagram of how such a Neural Network would be designed using a predefined Interface and Neuron Factory.

The Neural Network employs the Prototype Design Pattern [38], which allows created neuron types to be stored and accessed in a hash-table that is then referenced and accessed during run-time, as Freeman et al. [57] describe that it is easier to access and use for existing objects that have already been created and are resident in memory than it is to create and use new components during run-time. As a result, it is expected that this design pattern and its implementation will not only reduce the amount of system overhead associated with component creation but will also improve performance by reducing resource requirements while increasing response time.

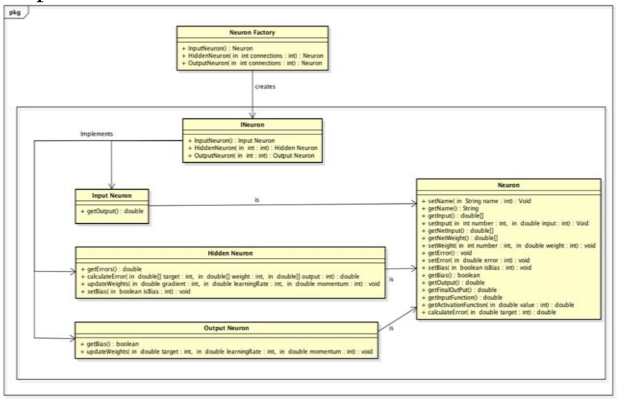

pkg
Neuron Factory
+ InputNeuron() : Neuron
+ HiddenNeuron(in int connections : int) : Neuron
+ OutputNeuron(in int connections : int) : Neuron
creates
INeuron
+ InputNeuron() : Input Neuron
+ HiddenNeuron(in int : int) : Hidden Neuron
+ OutputNeuron(in int : int) : Output Neuron
Implements
Input Neuron
+ getOutput() : double
is
Hidden Neuron
+ getErrors() : double
+ calculateError(in double[] target : int, in double[] weight : int, in double[] output : int) : double
+ updateWeights(in double gradient : int, in double learningRate : int, in double momentum : int) : void
+ setBias(in boolean isBias : int) : void
is
Output Neuron
+ getBias() : boolean
+ updateWeights(in double target : int, in double learningRate : int, in double momentum : int) : void
is
Neuron
+ setName(in String name : int) : Void
+ getName() : String
+ getInput() : double[]
+ setInput(in int number : int, in double input : int) : Void
+ getNetInput() : double[]
+ getNetWeight() : double[]
+ setWeight(in int number : int, in double weight : int) : void
+ getError() : void
+ setError(in double error : int) : void
+ setBias(in boolean isBias : int) : void
+ getBias() : boolean
+ getOutput() : double
+ getFinalOutPut() : double
+ getInputFunction() : double
+ getActivationFunction(in double value : int) : double
+ calculateError(in double target : int) : double

Figure 3 Neural Network Factory Pattern Class Diagram.

The Visualization Desktop Application functions as a framework, loading, invoking, and exposing methods in commonly created desktop application components, which when combined provide a graphical overview and representation of the Neural Network, allowing for parameter updates and network configurability. The Neural Network Framework depicted in figure 4 handles all processing and creational logic associated with neural networks, this component was developed to use the Front Controller Design Pattern as described by Freeman et al. [57] to provide a centralized request handling mechanism that will handle component authentication during run-time, which is used by the Desktop Application. Figure 4 also depicts several class diagrams, their methods, and their relationships to one another, providing a graphical representation of the proposed Desktop Application that has implemented the design pattern. However, the Desktop Application Framework will be used not only to create components, but also as a message distribution mechanism to facilitate collaboration between components, as well as to track and log requests made by components. However, because Desktop Application Components are referenced in run-time using public methods accessed via signature coupling via dynamic late binding [58], the Desktop Application Framework will be unable to distinguish between component types and will instead broad cast events and messages generated to all components, which will then be required to filter messages and only act on those messages specific to those components.

The implanted software artefact consists of a custom developed subroutine and code that can be plugged into the legacy financial banking system. As previously discussed, the Neural Network is the heart of the Desktop Application and Neural Interface and is based on a Perceptron Neural Network model, consisting of individual neuron objects, each with their own business logic, methods, and validation rules inherited from the base Neuron Object created using a Factory Design Pattern.

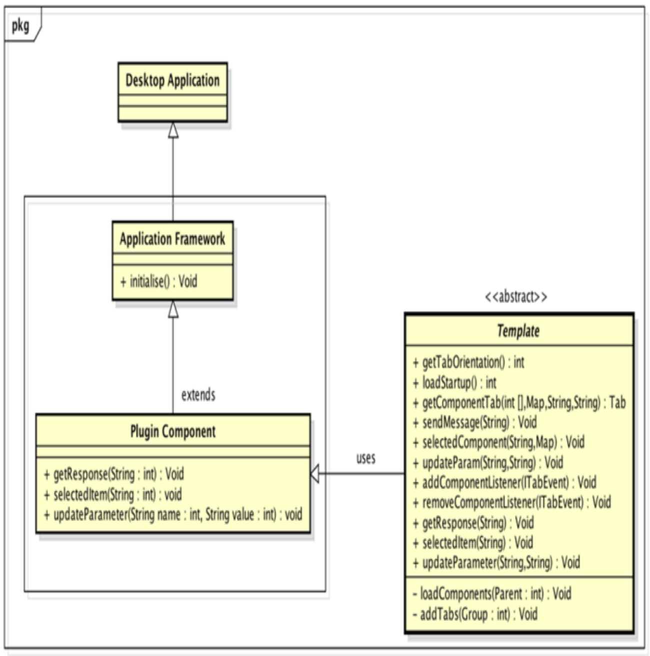
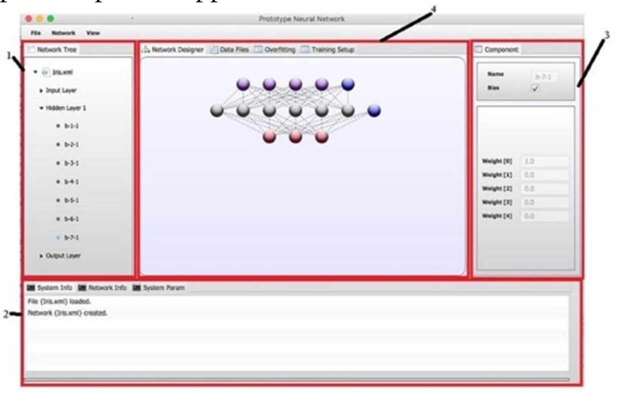

pkg
Desktop Application
Application Framework
+ initialise() : Void
extends
Plugin Component
+ getResponse(String : int) : Void
+ selectedItem(String : int) : void
+ updateParameter(String name : int, String value : int) : void
uses
<<abstract>>
Template
+ getTabOrientation() : int
+ loadStartup() : int
+ getComponentTab(int [],Map,String,String) : Tab
+ sendMessage(String) : Void
+ selectedComponent(String,Map) : Void
+ updateParam(String,String) : Void
+ addComponentListener(ITabEvent) : Void
+ removeComponentListener(ITabEvent) : Void
+ getResponse(String) : Void
+ selectedItem(String) : Void
+ updateParameter(String,String) : Void
- loadComponents(Parent : int) : Void
- addTabs(Group : int) : Void

Figure 4 Class Diagram for the Front Controller Pattern implemented by the Desktop Application

As previously stated, input vectors and computed output results are returned using a standard feed-forward mechanism common in Neural Networks, in which each vector passes through each layer and Neuron performing input summation followed by an activation function. The Sigmoid Activation function is used for each neuron in the hidden layer before passing vectors to the levels below it in this study.

However, in order to train the Neural Network to correctly identify data and classify information based on patterns, it is proposed that the network be trained using the standard backpropagation gradient descent – chain rule algorithm, which allows for a gradual reduction in the error rate via an iterative cycle process to eventually match the MSE rate as provided by the user. In this research, the standard backpropagation algorithm was used as it is easier to code and test but may be replaced by a more modern training method if the proof of concept is demonstrated.

The installed Visualization Desktop application serves as a framework for combining and displaying separate components that collectively comprise the Desktop Application, and it was implemented by using the Java language and by loading JAR files of pre-created components in run-time and displaying their information in a graphical format using a Tab per component approach.



Name: b-7-1
Bias: ✓
Weight [0]: 1.0
Weight [1]: 0.0
Weight [2]: 0.0
Weight [3]: 0.0
Weight [4]: 0.0

File (Iris.xml) loaded.
Network (Iris.xml) created.

Figure 5 Neural Network On-Line Training Cycle using Standard Backpropagation

Figure 5 shows the internal desktop framework logic used to upload component JAR files and invoke the components. While the blue section represents the general shared communication objects and protocol used to invoke exposed methods in components using the Template Design Pattern. The Singleton Design Pattern is used by the Desktop Application, which means that each component only has one instance of the Desktop Application running. Figure 6 depicts the architectural overview and workflow for creating and communicating plugin components, as well as the workflow and business logic for this design pattern and its implementation.

Because the Neural Network has a several different exposed methods compared to the collective components that act as plugin network configurable nodes, the section in yellow in figure 6 lists specific communication with the Neural Network.

Because software components are specifically developed as visual representations of a created neural network updated and displayed in a tabular format using the Desktop Application Framework as a common platform, components will contain parameterized display information describing where they will appear on the screen as well as when they will be displayed, depending on the workflow processes.

Figure 6 shows the possible locations where any component may appear based on a defined grid format coded into the Visualization Desktop Application Framework.

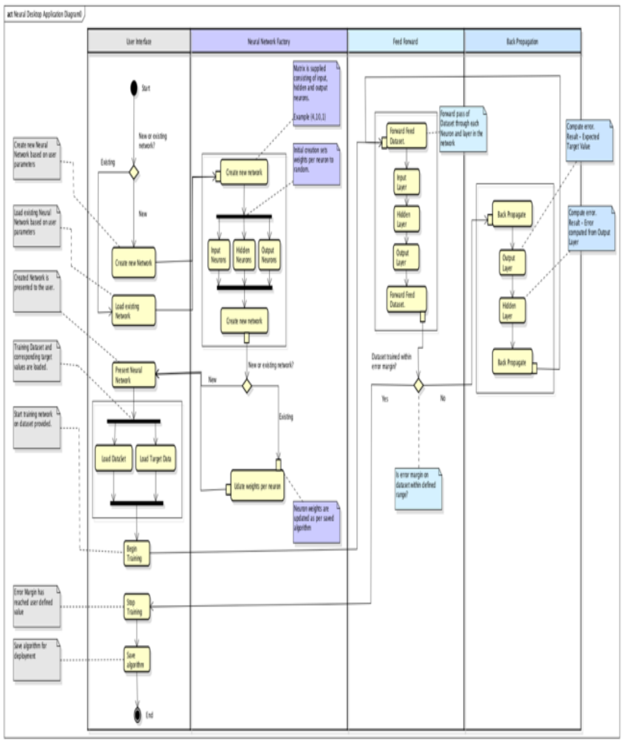

act Neural Desktop Application Diagram
User Interface
Neural Network Factory
Feed Forward
Back Propagation
Start
New or existing network?
Existing
New
Create new Neural Network based on user parameters
Load existing Neural Network based on user parameters
Created Network is presented to the user.
Training Dataset and corresponding target values are loaded.
Start training network on dataset provided.
Error Margin has reached user defined value
Save algorithm for deployment
Create new Network
Load existing Network
Present Neural Network
Load DataSet
Load Target Data
Begin Training
Stop Training
Save algorithm
End
Matrix is supplied consisting of input, hidden and output neurons.
Example (4,10,1)
Initial creation sets weights per neuron to random.
Create new network
Input Neurons
Hidden Neurons
Output Neurons
Create new network
New or existing network?
New
Existing
Udate weights per neuron
Neuron weights are updated as per saved algorithm
Forward Feed Dataset.
Forward pass of Dataset through each Neuron and layer in the network
Input Layer
Hidden Layer
Output Layer
Forward Feed Dataset.
Dataset trained within error margin?
Yes
No
Is error margin on dataset within defined range?
Back Propagate
Output Layer
Hidden Layer
Back Propagate
Compute error. Result - Expected Target Value
Compute error. Result - Error computed from Output Layer

Figure 6 Neural Network Framework – Feed Forward Workflow and System Log.

3 Results

The development for the Neural Network Framework, Neural Interface and Desktop plugin components was performed by using, Java JDK 1.8 on NetBeans 8.1 and the Desktop Application acting as a UI (User Interface) was developed using JavaFX 1.8. Unit and system testing was conducted by using JUnit (Junit.org, 2016), following a test driven development approach (TDD).

The results and findings presented in this section are based on system implementation, integration with a financial legacy systems, and test results obtained over a two-year period using datasets provided by a large commercial retailer (2014-2016). Mazars in South Africa independently audited and confirmed all results and findings.

An MLP (Multi-Layer Perceptron) neural network configuration consisting of different input neurons and different output neurons, with one hidden layer where each neuron will return a classification value, was used for the implementation. (Threshold Logic: Neuron responses less than 0.5 were treated as zero, while neuron responses greater than 0.5 are treated as one). The different configurations used are in figure 13.

While several dataset processing techniques are used in ANN training, Gallant [59] claims that the most commonly used splitting ratio for ANN using a single dataset for training, generalization, and validation is 60%, 20%, 20%, as shown in figure 7, where a network is:

- Trained on 60% of the data with both inputs and expected target values presented to the network, [60].
- Generalization, which accounts for 20% of a dataset, measures how well a network can generalize on previously unseen results [61] and serves as a performance indicator [62] by passing both input and target values.
- Validation data, which is made up of 20% of a training data set, is used to simulate unseen data that was not used in training or generalization verification and serves as a production environment simulation and final network performance indicator.

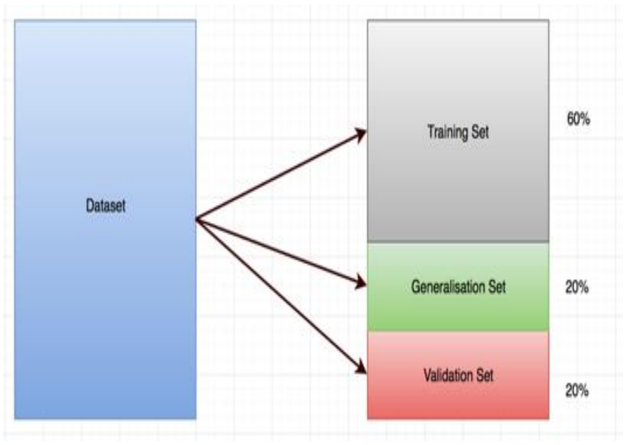


Figure 7 Standard dataset split used for training, generalization and validation.

Using the same method, our data will be divided into the following datasets based on the above ratios (where 56974 is the total dataset size) $\left(\frac{6974_{fraud}}{50000_{legitimate}}\right)$:

i. Training data = (56974 * 60%) = *34184*

ii. Validation data = (56974 * 20%) = *11395*

iii. Generalization data = (56974 * 20%) = *11395*

Note – *Percentage ratio outputs are rounded up to the nearest whole number.*

Similarly, Istook and Martinez [63] claim that "windowed momentum," also known as data shuffling, as shown in Figure 8, "achieved an average speed-up of 32% in convergence time" when compared to standard methods.

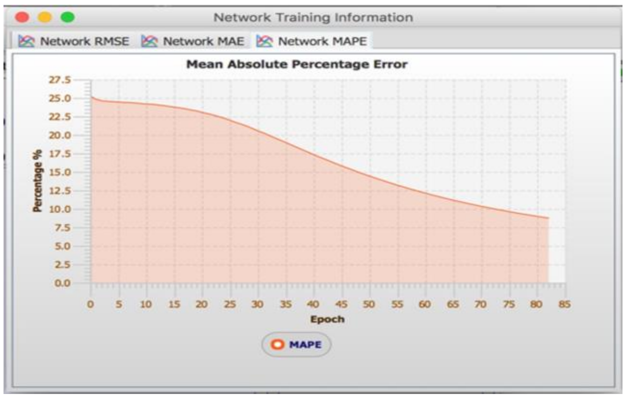


Figure 8 Data shuffling / windowed dataset

While several dataset processing techniques are used in ANN training, Gallant [25] claims that splitting a dataset is the most commonly used technique. While applying Ockham's Razor [66] to determine the number of input and output nodes on a network is relatively straightforward, [51], Livingstone and Luik, [67], warn that determining the exact number of hidden neurons frequently involves a trial-and-error approach in which a network is trained for a number of times using variant parameters.

Winston [68] adds that momentum, learning rates, and MSE (Mean Squared Error) variations may have an adverse effect on network performance and convergence rates. For this reason, the research uses a trial-and-error approach, training the ANN on the same datasets, using identical initial weighted connections, and then selecting the one with the best performance. convergence rates and least amount of overfitting as shown in figure 9.

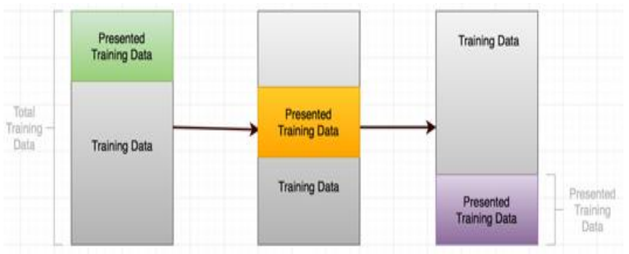


Figure 9 Training samples using different network configurations

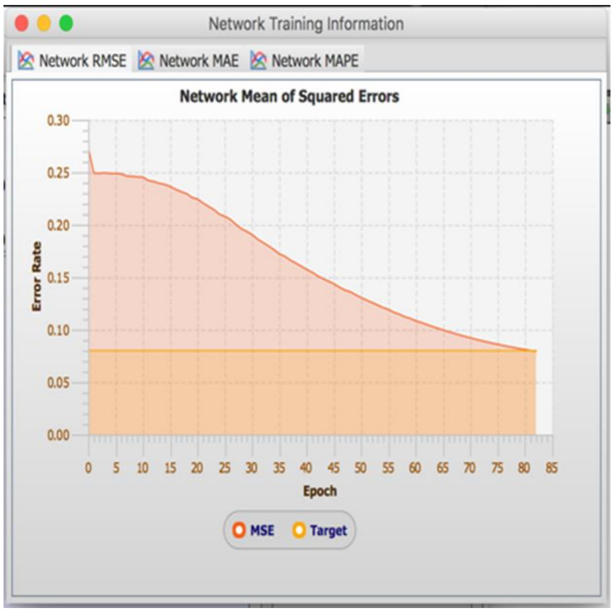


Figure 10 depicts a decrease in MAE (Mean Absolute Error) rates during the same training cycle, whereas Figure 11 depicts a decrease in MAPE (Mean Absolute Percentage Error) rates during the training cycle.

Figure 10 MAE rates during training using network configuration 5

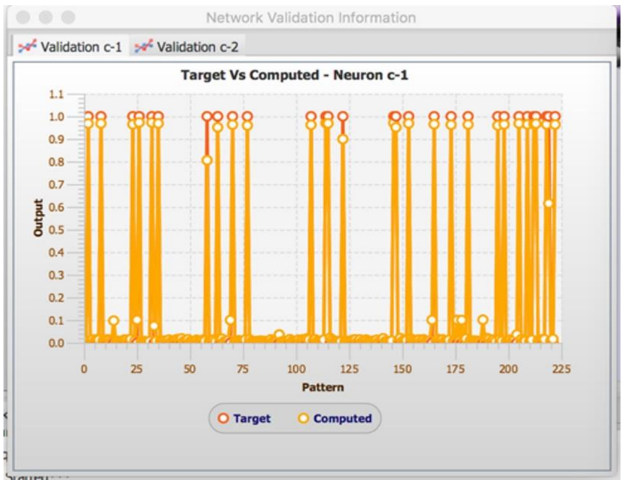
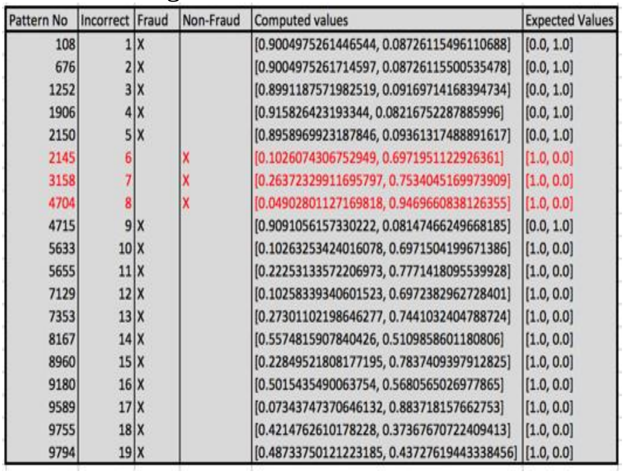


Figure 11 MAPE rates during training using network configuration 5

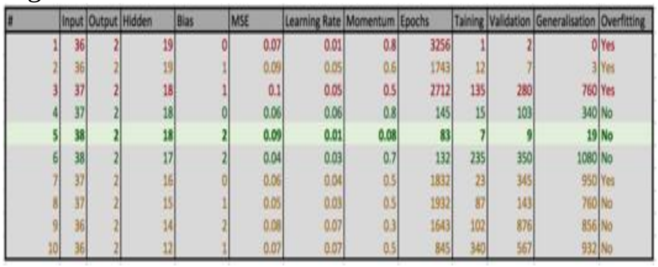
As described and shown in figure 12, the ANN configuration performed best with the integration of the financial legacy system on the datasets provided by network configuration number 5. This because this configuration not only produces shorter training cycles but also matches a better fitting curve with fewer errors during validation and generalization, reporting a total of 19 incorrect results out of 11395 as shown in figure 12.



Pattern No	Incorrect	Fraud	Non-Fraud	Computed values	Expected Values
108	1	X		[0.9004975261446544, 0.08726115496110688]	[0.0, 1.0]
676	2	X		[0.9004975261714597, 0.08726115500535478]	[0.0, 1.0]
1252	3	X		[0.8991187571982519, 0.09169714168394734]	[0.0, 1.0]
1906	4	X		[0.915826423193344, 0.08216752287885996]	[0.0, 1.0]
2150	5	X		[0.8958969923187846, 0.09361317488891617]	[0.0, 1.0]
2145	6		X	[0.1026074306752949, 0.6971951122926361]	[1.0, 0.0]
3158	7		X	[0.26372329911695797, 0.7534045169973909]	[1.0, 0.0]
4704	8		X	[0.04902801127169818, 0.9469660838126355]	[1.0, 0.0]
4715	9	X		[0.9091056157330222, 0.08147466249668185]	[0.0, 1.0]
5633	10	X		[0.10263253424016078, 0.6971504199671386]	[1.0, 0.0]
5655	11	X		[0.22253133572206973, 0.7771418095539928]	[1.0, 0.0]
7129	12	X		[0.10258339340601523, 0.6972382962728401]	[1.0, 0.0]
7353	13	X		[0.27301102198646277, 0.7441032404788724]	[1.0, 0.0]
8167	14	X		[0.5574815907840426, 0.5109858601180806]	[1.0, 0.0]
8960	15	X		[0.22849521808177195, 0.7837409397912825]	[1.0, 0.0]
9180	16	X		[0.5015435490063754, 0.5680565026977865]	[1.0, 0.0]
9589	17	X		[0.07343747370646132, 0.883718157662753]	[1.0, 0.0]
9755	18	X		[0.4214762610178228, 0.37367670722409413]	[1.0, 0.0]
9794	19	X		[0.48733750121223185, 0.43727619443338456]	[1.0, 0.0]

Figure 12 Incorrectly classified transactions based on best training curve with least overfitting

Similarly, graphical results (shown in Figure 13) returned by the system during generalization, where target values are not presented during result computations but are matched against system output results, revealed findings similar to those shown in Figure 12.



#	Input	Output	Hidden	Bias	MSE	Learning Rate	Momentum	Epochs	Taining	Validation	Generalisation	Overfitting
1	36	2	19	0	0.07	0.01	0.8	3256	1	2	0	Yes
2	36	2	19	1	0.09	0.05	0.6	1743	12	7	3	Yes
3	37	2	18	1	0.1	0.05	0.5	2712	135	280	760	Yes
4	37	2	18	0	0.06	0.06	0.8	145	15	103	340	No
5	38	2	18	2	0.09	0.01	0.08	83	7	9	19	No
6	38	2	17	2	0.04	0.03	0.7	132	235	350	1080	No
7	37	2	16	0	0.06	0.04	0.5	1832	23	345	950	Yes
8	37	2	15	1	0.05	0.03	0.5	1932	87	143	760	No
9	36	2	14	2	0.08	0.07	0.3	1643	102	876	856	No
10	36	2	12	1	0.07	0.07	0.5	845	340	567	932	No

Figure 13 Training samples using different network configurations

4 Discussion

According to the findings of this study, the visualization and data mining ANN artefact developed for this paper, which is capable of dynamically creating a variety of user defined Neural Network architectures varying in network topology, connectional patterns, and size, was able to successfully train, validate, and generalize on datasets and patterns provided by the case study company, reporting estimated accuracy levels.

However, while the network's predictive capability appears to have a high performance, this is consistent with research results reported by Patidar and Sharma [9], who conducted a similar study on credit card fraud and reported accuracy levels of 98 percent. Similarly, Ghosh and Reilly [40], who conducted a feasibility study at Mellon Bank using customer-supplied datasets covering lost cards, stolen cards, application fraud, counterfeit fraud, mail order fraud, and NRI (Non-Received Issue) fraud, concluded that “the network detected significantly more fraud accounts (an order of magnitude more) with significantly fewer false positives (reduced by a factorial approach)”. However, none of the previous studies addressed the issue of integration with legacy financial information systems that is the main contribution of this rearch.

The research results show that predictive estimator accuracy levels appear relatively high due to data correlation issues [69], statistical bias [65], indicators, or common overfitting (overfitting, 2016), all of which occur in Perceptron-based Neural Network models. Overfitting, as described by Duro and Reyes [42], can lead to poor predictive results on unseen data, whereas predictive ratios between seen and unseen results show a variance of 7, 9, and 19 incorrectly reported transactions (on algorithm 5, shown in Table 3) between training, validation, and generalization datasets used, giving a gradient= (y 2-y 1)/ (x 2-x 1) = (19-9) (9-7) = 10/2=5. That is, while predictions on unseen results appear to be poorer, the deviation is only 5%.

According to Dezhong Peng et al. [50] if a statistical bias is introduced, model design may incorrectly under or overcompensate for missing variables by under or overestimating connectional weights, resulting in variable connectional weights that become inadvertently large to compensate for incorrectly correlated data,]. However, as shown in Figure 14, in our research, neuron weights are close to zero, leading us to believe that statistical biases did not contribute to data and result findings in this case.

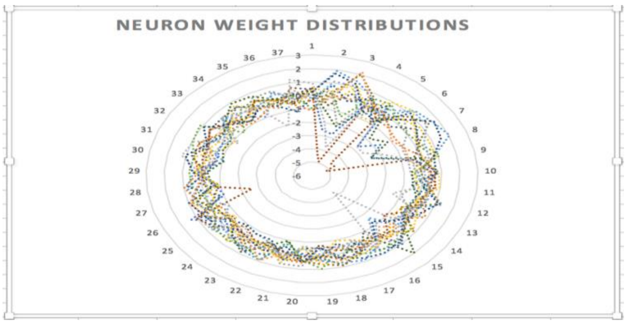


Figure 14 Neuron weight distributions.

Credit card fraud data, on the other hand, can be described as "skewed distributions" by Stolfo et al. [34] because the number of fraudulent transactions is statistically significantly lower than the total volume of transactions. Such skewed distributions are expected to appear within datasets even after normalization, using a Gaussian distribution function, expressed as p(x)=Ae^(-λ (x-a)^2), specifically designed to deal with variables that are “inherently positive or strongly skewed” or classification overlaps (Fawcett and Provost, 1997) as in Figure 15, representing generalized data consisting of 11395 t. Figure 16 depicts computed output responses, whereas Figure 17 depicts actual expected target values.

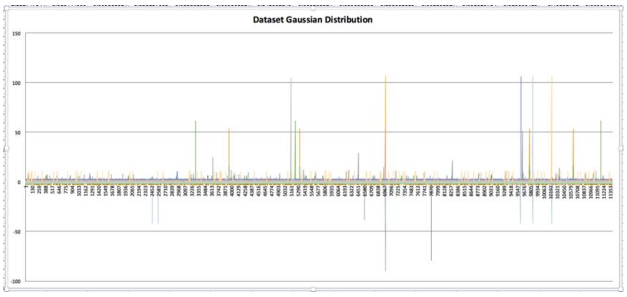


Figure 15 Gaussian Data Distribution

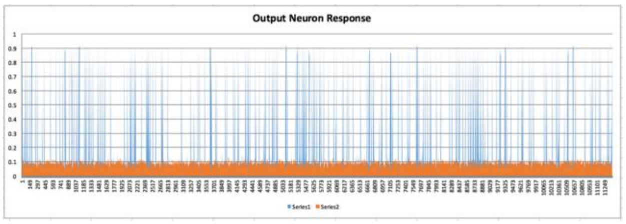


Figure 16 Computed output Neuron responses.

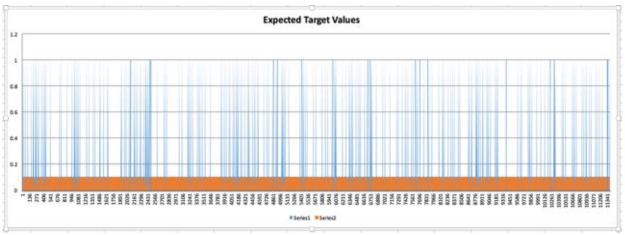


Figure 17 Expected Target Values.

According to the study's findings, the IT Artefact developed for this paper, which is capable of dynamically creating a variety of user defined Neural Network architectures varying in network topology, connectional patterns, and size, was able to successfully train, validate, and generalize on datasets and patterns provided by the company used for the case study with estimated accuracy levels of 98.33.

As a result, the research shows that Perceptron-based Neural Networks can successfully identify, classify, and predict credit card fraud within the datasets provided, levels of accuracy are dependent on network size, training patterns provided, and data quality. While the system correctly identifies the vast majority of transactions, it should be noted that in some cases, transactional data deviates from normal distribution curves, and such anomalies must be considered when developing fraud prevention strategies. A limitation to consider is that the research and findings are limited to a single customer and dataset, and results may differ depending on datasets, an organization's geographic area of operation, or operational sector within the industry.

4 Conclusions

As demonstrated in our paper, data mining can be used to address both business and technical challenges. Data mining provides powerful, effective, and extremely useful methods for processing large amounts of data, quickly and accurately completing data analysis, and providing real solutions to help businesses move forward with their strategies, operations, and solutions for their clients in the following major areas.

The main goal of this paper was to demonstrate how to use techniques for pattern recognition and credit card data classification commonly found in the financial and retail industries to identify fraudulent credit card transactions using a flexible and configurable Artificial Neural Network (ANN) software component that can plug directly into a legacy system. This technique may be useful in the reengineering of legacy systems in the banking industry.

First, using datasets provided by company for the case study, the visualization and data mining Artificial Neural Network (ANN) modules were developed and they were able to successfully identify, predict, and classify fraudulent credit card transactions with an accuracy level of 98 percent (Table 1), but training on large datasets proved difficult due to data availability.

Table 1 Accuracy level of ANN

$\sigma = \sqrt{\left[\frac{\sum d^2}{n-1}\right]}$
$skewness = \frac{fraudulent\ transactions}{dataset\ size} = 0.1224 = \frac{6975}{56973}$
$accurancy = \left(\frac{generalisation\ dataset}{incorrect\ transactions}\right) * 100 = 99.83 = \left(\frac{11394}{19}\right) * 100$
$error\ correction = (accuracy\ levels * margin\ of\ error) \approx 97.83 \approx (99.83 * 2\%) - 99.83$

While software design patterns should be used to solve commonly occurring architectural problems found in the industry [38], it is important to avoid overengineering the system architecture or coding methods for the sake of using patterns. The proposed solution was capable of resolving technical and business challenges using step-by-step approaches, as evidenced by our analytics.

While determining the number of input and output neurons is usually straightforward often correlated with the number of data points in a dataset or linked to the number of classifiers required, determining the exact number of hidden neurons, learning rate, momentum [49], and stopping criteria remains a trial and error process, according to the study.

Second, this paper demonstrated that levels of accuracy, speed of convergence, and predictor estimations within any Neural Network are linked to the quality and quantity [68] of data provided, and that while a Neural Network may perform well during training and validation sessions, generalization issues may arise as a result of overfitting when creating networks that are inherently too complicated. Also, underfitting occurs when a network structure is too light in weight, resulting in non-convergence problems, which are also considered a trial-and-error process.

Third, while credit card fraud can be predicted and identified with some accuracy as a non-linear data regression and classification problem [66], there are transactions that fall outside of standard distribution and Gaussian curves, resulting in false positives and Visa-Versa, and such anomalies must be taken into consideration.

While there is a substantial amount of literature available within the academic community covering statistical methods, AI, Support Vector Machines (SVM), Nave Bayes (NB) networks [12], K-Nearest Neighbor (KNN) algorithms [70], and Bayesian systems used in regression and classification data problems covering credit card fraud and levels of accuracy. There is no evidence of research that focuses on the integration of an ANN solution for legacy systems for the banking industry.

Fourth, this paper demonstrated data mining by providing adequate levels of information and architectural design used to develop an object-oriented Java-based data mining and visualization module that can be integrated with financial legacy systems which is the main contribution of this research. The system design and application architecture presented in this paper should provide sufficient detail to aid in the development of future scalable, modular object-oriented neural network systems that can be integrated with any type of legacy financial system. However, it should be noted that the design and implementation of this system is a prototype with limited capability, employing only a single training algorithm, but it could easily be improved in the future to include new and more modern learning rules and algorithms. Backpropagation was the training algorithm used.

Fifth, the data mining presented in this paper contributed to knowledge sharing by expanding on existing knowledge with a study that shows how to implement an ANN solution for fraud detection for legacy financial systems with a special emphasis on Perceptron modelled Neural Networks and credit card data classification problems. However, because the data was gathered from a single case study operating in a specific geographic location (South Africa) and commercial sector (retail), the results, findings, and model evaluation are limited to what our case study provided.

Finally, we successfully demonstrated the efficacy of data mining by combining supervised and unsupervised learning methods. In other words, our research contributions corresponded exactly to Mortensen's proposal [71]. It contained three components: technologies, quantitative methods, and decision-making. Our proposed software solution could seamlessly combine the results of AI, information systems, and operational research. As a result, we were able to meet our goal of detecting credit card fraud.

While the IT Artefact developed for this paper, modelled after the Perceptron [1,] is capable of learning from and predicting credit card fraud with sufficient accuracy to warrant further academic and commercial application exploration, neural network models that use on-line supervised training as a learning rule are limited to detection, prediction, and classification of known and predicted events. According to Phansalkar and Sastry [32], detecting new or novel transactional types would necessitate reinforced training methods or application on unsupervised learning systems.

The research used an MLP (Multi-layered Perceptron) networks with a single hidden layer, this according to Ackley, Hinton, and Sejnowski [72], can solve the majority of non-linear data problems such as credit card fraud detection. Deep Learning [31] allows for the training, classification, and development of more complex systems using significantly larger datasets, which frequently oversaturate the connectional weights of MLP (Multi-layered Perceptron)-based networks with a single hidden layer, this tool can be explored in future research. Also, for future research, a study could be conducted to determine the saturation points of connectional weights in single hidden layered MLPs versus multi-layered networks, because neurons within a network statistically follow distribution curves [68] and may miss or interpret data falling outside such ranges.

Data mining and visualization adaptations could be also part of our future research. The artefact developed for this paper to implement self-mapping Neural Networks, SOM and SOFM types, allowing for unsupervised training methods by applying competitive learning strategies using a neighborhood function and a study of either Kohonen [75] or morphogenesis models [74], could be evaluated and compared to traditional backpropagation [76] learning rules. Some other relevant studies can be found in [72] and [73].

References

[1] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), pp.386-408

[2] Lamb, J. (2008). Legacy systems continue to have a place in the enterprise. [online] ComputerWeekly. Available at: http://www.computerweekly.com/feature/Legacy-systems-continue-to-have-a-place-in-the-enterprise [Accessed 29 Feb. 2016].

[3] level2kernel, (2016). How EMV (Chip & PIN) Works. [online] How EMV (Chip & PIN) Works - Transaction Flow Chart. Available at: https://www.level2kernel.com/flow-chart.html [Accessed 4 Mar. 2016].

[4] Barclaycard, (2016). How to use Contactless Payment Cards | Barclaycard. [online] Barclaycard.co.uk. Available at: https://www.barclaycard.co.uk/personal/credit-cards/using-contactless [Accessed 4 Mar. 2016].

[5] Nicholls, C. (2013). Are Verified by Visa and MasterCard SecureCode Conversion Killers?. [online] Practical Ecommerce. Available at: http://www.practicalecommerce.com/articles/4059-Are-Verified-by-Visa-and-MasterCard-SecureCode-Conversion-Killers- [Accessed 4 Mar. 2016].

[6] Shankland, S. (2015). Visa says new app will cut credit-card travel troubles. [online] CNET. Available at: http://www.cnet.com/news/visa-says-new-app-will-cut-credit-card-travel-troubles/ [Accessed 29 Feb. 2016].

[7] Zetter, K. (2015). That Big Security Fix for Credit Cards Won’t Stop Fraud. [online] WIRED. Available at: http://www.wired.com/2015/09/big-security-fix-credit-cards-wont-stop-fraud/ [Accessed 29 Feb. 2016].

[8] Davis, M. (2000). Continuous improvement and return on investment. Minneapolis: Capella University, pp.13-19.

[9] Patidar, R. and Sharma, L. (2011). Credit Card Fraud Detection Using Neural Network. International Journal of Soft Computing and Engineering (IJSCE), 1(NCAI2011), pp.32-38
[10] Nigrini, M. (2011). Forensic analytics. Hoboken, N.J.: Wiley
[11] Palshikar, G. (2002). The Hidden Truth – Frauds and Their Control: A Critical Application for Business Intelligence. Intelligent Enterprise, 5(9), pp.46–51.
[12] Zhang, R. and Bivens, A. (2007). Comparing the use of bayesian networks and neural networks in response time modeling for service-oriented systems. Proceedings of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches - SOCP '07, pp.67-74.
[13] Stolfo, S., Fan, D., Lee, W., Prodromidis, A. and Chan, P. (1997). Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results. Proc. AAAI Workshop AI Methods in Fraud and Risk Management, pp.83-90.
[14] Joshi, A., Ramakrishman, N., Houstis, E. and Rice, J. (1997). On neurobiological, neuro-fuzzy, machine learning, and statistical pattern recognition techniques. IEEE Trans. Neural Netw., 8(1), pp.18-31.
[15] Gómez, J. A., Arévalo, J., Paredes, R., & Nin, J. (2018). End-to-end neural network architecture for fraud scoring in card payments. Pattern Recognition Letters, 105, 175-181.
[16] Wang, C., Wang, Y., Ye, Z., Yan, L., Cai, W., & Pan, S. (2018, August). Credit card fraud detection based on whale algorithm optimized bp neural network. In 2018 13th International Conference on Computer Science & Education (ICCSE) (pp. 1-4). IEEE.
[17] Zhang, Z., Zhou, X., Zhang, X., Wang, L., & Wang, P. (2018). A model based on convolutional neural network for online transaction fraud detection. Security and Communication Networks, 2018.
[18] Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. Expert Systems with Applications, 100, 234-245
[19] Davenport, T.H. and Harris, J.G., (2007). Competing on analytics: The new science of winning. Harvard Business Press.
[20] Massa, D., & Valverde, R. (2014). A fraud detection system based on anomaly intrusion detection systems for e-commerce applications. Computer and Information Science, 7(2), 117.
[21] Talla, M. R., & Valverde, R. (2012). Data oriented and Process oriented Strategies for Legacy Information Systems Reengineering. ACEEE International Journal on Information Technology, 2(1), 47-51
[22] Talla, M. R., & Valverde, R. (2012). Data oriented and Process oriented Strategies for Legacy Information Systems Reengineering. ACEEE International Journal on Information Technology, 2(1), 47-51
[23] Valverde, R., & Talla, M. (2017). Reengineering of Legacy Systems into Supply Chain Systems: Traditional Data Oriented versus Process Oriented Approaches. International Journal of Organizational and Collective Intelligence (IJOCI), 7(2), 1-12.
[24] Picton, P. (2000). Neural Networks. 2nd ed. New York: Palgrave, pp.1-49, 155-165.
[25] Gallant, S. I., & Gallant, S. I. (1993). Neural network learning and expert systems. MIT press.
[26] Rumelhart, D. and McClelland, J. (1986). Parallel distributed processing. Cambridge (Mass.): MIT Press
[27] Chan, P., Fan, W., Prodromidis, A. and Stolfo, S. (1999). Distributed data mining in credit card fraud detection. IEEE Intell. Syst., 14(6), pp.67-74.
[28] Dorronsoro, J., Ginel, F., Sgnchez, C. and Cruz, C. (1997). Neural fraud detection in credit card operations. IEEE Trans. Neural Netw., 8(4), pp.827-834
[29] Kokkinaki, A. (1997). On atypical database transactions: identification of probable frauds using machine learning for user profiling. Proceedings 1997 IEEE Knowledge and Data Engineering Exchange Workshop, pp.107-113.
[30] Bryson, A., Ho, Y. and Siouris, G. (1979). Applied Optimal Control: Optimization, Estimation, and Control. IEEE Transactions on Systems, Man, and Cybernetics, 9(6), pp.366-367.
[31] Alpaydin, E. (2010). Introduction to machine learning. Cambridge, Mass.: MIT Press, p.250.
[32] Phansalkar, V. and Sastry, P. (1994). Analysis of the back-propagation algorithm with momentum. IEEE Trans. Neural Netw., 5(3), pp.505-506.
[33] Cortes, C. and Pregibon, D. (2001). Signature-Based Methods for Data Streams. Data Mining and Knowledge Discovery, 5, pp.167–182.
[34] Stolfo, S., Fan, D., Lee, W., Prodromidis, A. and Chan, P. (1997). Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results. Proc. AAAI Workshop AI

Methods in Fraud and Risk Management, pp.83-90.

[35] Simon, P. (2013). Too Big to Ignore: The Business Case for Big Data. Hoboken, New Jersey: John Wiley & Sons, p.89.

[36] Fawcett, T. and Provost, F. (1997). Adaptive Fraud Detection. Data Mining and Knowledge Discovery, 1(3), pp.291-316.

[37] Mansfield, S. (2015). Number of attacks against retailers falls, but the severity and sophistication increase. Computer Fraud & Security, 2015(1), pp.1-3

[38] Fowler, M. (2003). Patterns of enterprise application architecture. Boston: Addison-Wesley.

[39] Twomey, J. and Smith, A. (1995). Performance measures, consistency, and power for artificial neural network models. Mathematical and Computer Modelling, 21(1-2), pp.243-258.

[40] Ghosh, and Reilly, (1994). Credit card fraud detection with a neural-network. Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences HICSS-94, 3(4-7 Jan), pp.621 - 630.

[41] White, H. (1990). Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. Neural Networks, 3(5), pp.535-549.

[42] Duro, R. and Reyes, J. (1999). Discrete-time backpropagation for training synaptic delay-based artificial neural networks. IEEE Trans. Neural Netw., 10(4), pp.779-789.

[43] Falas, T. and Stafylopatis, A. (1999). The impact of the error function selection in neural network-based classifiers. IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339), 3, pp.1799 - 1804.

[44] Hyndman, R. and Koehler, A. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting, 22(4), pp.679-688.

[45] Saravanan, K. and Sasithra, S. (2014). Review on Classification Based on Artificial Neural Networks. IJASA, 2(4), pp.11-18.

[46] Tino, P., Cernansky, M. and Benuskova, L. (2004). Markovian Architectural Bias of Recurrent Neural Networks. IEEE Trans. Neural Netw., 15(1), pp.6-15.

[47] Sinčák, P. (2002). Intelligent technologies--theory and applications. Amsterdam: IOS Press, pp.19-23.

[48] Balabin, R., Safieva, R. and Lomakina, E. (2007). Comparison of linear and nonlinear calibration models based on near infrared (NIR) spectroscopy data for gasoline properties prediction. Chemometrics and Intelligent Laboratory Systems, 88(2), pp.183-188.

[49] Liyi, Z., Ting, L. and Jingyu, Z. (2009). Analysis of Momentum Factor in Neural Network Blind Equalization Algorithm. 2009 WRI International Conference on Communications and Mobile Computing, 1, pp.345 - 348.

[50] Dezhong Peng, Zhang Yi, Yong Xiang, and Haixian Zhang, (2012). A Globally Convergent MC Algorithm With an Adaptive Learning Rate. IEEE Trans. Neural Netw. Learning Syst., 23(2), pp.359-365.

[51] Tetko, I., Livingstone, D. and Luik, A. (1995). Neural network studies. 1. Comparison of overfitting and overtraining. Journal of Chemical Information and Modeling, 35(5), pp.826-833.

[52] Leinweber, D. (2007). Stupid Data Miner Tricks. Investing, 16(1), pp.15-22.

[53] Jean, J. and Jin Wang, (1994). Weight smoothing to improve network generalization. IEEE Trans. Neural Netw., 5(5), pp.752-763.

[54] Karnin, E. (1990). A simple procedure for pruning back-propagation trained neural networks. IEEE Trans. Neural Netw., 1(2), pp.239-242.

[55] Nan-Ying Liang, Guang-Bin Huang, Saratchandran, P. and Sundararajan, N. (2006). A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. IEEE Trans. Neural Netw., 17(6), pp.1411-1423.

[56] Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. Neural Networks, 11(4), pp.761-767.

[57] Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). Head first design patterns. " O'Reilly Media, Inc.".

[58] Deitel, P. and Deitel, H. (2012). Java. 9th ed. Upper Saddle River, N.J.: Prentice Hall, p.13.

[59] Gallant, S. I., & Gallant, S. I. (1993). Neural network learning and expert systems. MIT press.

[60] LeCun, Y., Bottou, L., Orr, G. and Müller, K. (1998). Efficient BackProp. Lecture Notes in Computer Science, 1524, pp.9-50.

[61] Leinweber, D. (2007). Stupid Data Miner Tricks. Investing, 16(1), pp.15-22.

[62] Duro, R. and Reyes, J. (1999). Discrete-time backpropagation for training synaptic delay-based artificial neural networks. IEEE Trans. Neural Netw., 10(4), pp.779-789.

[63] ISTOOK, E. and MARTINEZ, T. (2002). IMPROVED BACKPROPAGATION LEARNING IN NEURAL NETWORKS WITH WINDOWED MOMENTUM. Int. J. Neur. Syst., 12(03n04), pp.303-318.
[64] Timmerman, A. (1997). Neural networks in finance and investing. Using artificial intelligence to improve realworld performance. International Journal of Forecasting, 13(1), pp.144-146.
[65] Reid, S. (2014). 10 Misconceptions about Neural Networks. [online] Turing Finance. Available at: http://www.turingfinance.com/misconceptions-about-neural-networks/ [Accessed 6 Mar. 2016].
[66] Oreskes, N., Shrader-Frechette, K. and Belitz, K. (1994). Verification, Validation, and Confirmation of Numerical Models in the Earth Sciences. Science, 263(5147), pp.641-646.
[67] Tetko, I., Livingstone, D. and Luik, A. (1995). Neural network studies. 1. Comparison of overfitting and overtraining. Journal of Chemical Information and Modeling, 35(5), pp.826-833.
[68] Winston, P. (2010). 6.034 Artificial Intelligence, Fall 2010.. [online] Massachusetts Institute of Technology: MIT OpenCourseWare. Available at: http://ocw.mit.edu [Accessed 7 Mar. 2016].
[69] Auer, P., Burgsteiner, H. and Maass, W. (2008). A learning rule for very simple universal approximators consisting of a single layer of perceptrons. Neural Networks, 21(5), pp.786-795.
[70] Tu, J. V. (1996). Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. Journal of clinical epidemiology, 49(11), 1225-1231.
[71] Mortenson, M.J., Doherty, N.F. and Robinson, S., 2015. Operational research from Taylorism to Terabytes: A research agenda for the analytics age. European Journal of Operational Research, 241(3), pp.583-595.
[72] Wei Yang, Yi Chai, Jie Zheng, Jie Liu, Intelligent Diagnosis Technology of Wind Turbine Drive System based on Neural Network, WSEAS Transactions on Circuits and Systems, pp. 289-296, Volume 19, 2020
[73] Alena Vagaská, Peter Michal, Ivo Bukovský, Miroslav Gombár, Ján Kmec, The Application of Neural Networks to Control Technological Process, WSEAS Transactions on Circuits and Systems, pp. 147-153, Volume 18, 2019
[74] Ackley, D., Hinton, G. and Sejnowski, T. (1985). A Learning Algorithm for Boltzmann Machines*. Cognitive Science, 9(1), pp.147-169.
[75] Auer, P., Burgsteiner, H. and Maass, W. (2008). A learning rule for very simple universal approximators consisting of a single layer of perceptrons. Neural Networks, 21(5), pp.786-795.
[76] Bishop, C. (2006). Pattern recognition and machine learning. New York: Springer, p.194.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0
https://creativecommons.org/licenses/by/4.0/deed.en_US

View publication stats