Q-Credit Card Fraud Detector for Imbalanced Classification using Reinforcement Learning

Luis Zhinin-Vera[1][a], Oscar Chang[1][b], Rafael Valencia-Ramos[1][c], Ronny Velastegui[1][d], Gissela E. Pilliza[1][e] and Francisco Quinga Socasi [1][f]

[1]*School of Mathematical and Computational Sciences, Yachay Tech University, 100650, Urcuqui, Ecuador*
luis.zhinin@yachaytech.edu.ec, luis1997@live.com.ar

Keywords: agents, credit card fraud detector, deep learning

Abstract: Every year, billions of dollars are lost due to credit card fraud, causing huge losses for users and the financial industry. This kind of illicit activity is perhaps the most common and the one that causes most concerns in the finance world. In recent years great attention has been paid to the search for techniques to avoid this significant loss of money. In this paper, we address credit card fraud by using an imbalanced dataset that contains transactions made by credit card users. Our *Q-Credit Card Fraud Detector* system classifies transactions into two classes: genuine and fraudulent and is built with artificial intelligence techniques comprising Deep Learning, Auto-encoder, and Neural Agents, elements that acquire their predicting abilities through a Q-learning algorithm. Our computer simulation experiments show that the assembled model can produce quick responses and high performance in fraud classification.

1 INTRODUCTION

Currently, fraud is the number one enemy in the business world. It affects industries and organizations and accounts for the big money invested in fraud prediction researching. The constant grow of this problem has strongly promoted the development of new technologies to counteract fraudsters. The last advances in credit card fraud detection include top technologies themes such as Artificial Neural Network (ANN), Deep Learning and Intelligent Agents. In particular the implementation of agents has become important since it produces effective, quick acting monitoring of credit card fraud transactions, reducing the risk of fraud or other financial traps that could signify losses. A quick response is essential because fraudsters are constantly creating new elaborated treachery mechanisms.

In terms of a robust functional system the main goal is to detect the highest possible number of fraudulent transactions using a finite dataset, in our case treated by Principal Component Analysis (PCA) approaches to anonymize the user and minimizes/maximize data correlation. Since frauds occurs with more frequency than regular transactions, databases are always imbalanced. This paper develops a fraud detection methodology that resolves the problem of imbalanced classification by combining the processing capacities of neural agents and Q-learning, establishing a promising way to satisfy quick acting and high precision requirement. The Reinforcement Learning method slightly outperforms neural networks while a similar representation is used (Wiering et al., 2011).

2 RELATED WORK

2.1 Credit Card Fraud Detection

An approach called *Long Short-term Memory Recurrent Neural Network (LSTM)* is used. Authors implement an ANN for detecting credit card fraud, taking into account sequences of transactions occurred in the past, in order to determine whether a new transaction is legitimate or fraudulent (Wiese and Omlin, 2009).

Checking the usage patterns of a user in previous transactions to detect a credit card frauds is suggested

[a] https://orcid.org/0000-0002-6505-614X
[b] https://orcid.org/0000-0002-4336-7545
[c] https://orcid.org/0000-0002-1036-1817
[d] https://orcid.org/0000-0001-8628-9930
[e] https://orcid.org/0000-0001-6386-9254
[f] https://orcid.org/0000-0003-3213-4460

(Dighe et al., 2018). They compare the usage pattern and current transaction, to classify it as either fraud or a legitimate transaction. Among the techniques implemented are KNN, Naïve Bayes, CFLANN, M-Perceptron and DTrees.

Credit cards frauds have no constant patterns is stated (Pumsirirat and Yan, 2018). Therefore, the use of an unsupervised learning is necessary. They take account that the frauds are committed once through online mediums and then the techniques change. To solve this issue, they implement a deep Auto-encoder model and a restricted Boltzmann machine, that can reconstruct normal transactions to find anomalies in the patterns.

An intelligent agent can obtain a high rate of fraud transaction with low false alarm rate, providing a convenient way to detect frauds (Chukwuneke, 2018). Their implementation of the intelligent agent is focus on detect the fraud when transaction is in progress, taking into account the costumers pattern, and any deviation from the regular pattern is considered to the fraudulent transaction.

2.2 Imbalanced Classification

Research work in imbalanced data classification is focused on two levels: data level and algorithmic level. In data level, the objective is balance the class distribution by manipulating the training samples, taking into account the over-sampling minority class, the under sampling majority class and their combinations. The authors takes into account that the oversampling can lead to overfitting while under-sampling lose valuable information on the majority class. On the other hand, the objective of the algorithmic level methods, is increase the importance of minority class by improving algorithms by decision threshold adjustment, cost-sensitive learning and ensemble learning (Lin et al., 2019). An alternative loss function in deep neural network that can capture the classification errors from both minority class and majority class is established (Wang et al., 2016). Extracting hard samples of minority classes and improved the bootstrapping sampling algorithm which ensure the training data in each mini-batch, by batch-wise optimization with Class Rectification Loss function (Dong et al., 2019).

2.3 Reinforcement Learning in Classification

Recently, the deep reinforcement learning has had excellent results, because it can assist classifiers to learn important features or select good instances from noisy data. The authors understand the classification task as sequential decision-making process, that uses a multiple agents interacting with the environment to obtain the optimal policy of classification. However, the interaction between agents and environment, generate an extremely high time complexity (Lin et al., 2019). Establishing a deep reinforcement learning model divided into instance selector and relational classifier, to learn the relationship classification in noisy text data. The instance selector part, implements an agent selects high quality sentence from noisy data while the relational classifier part learns from the previous selected data and give a reward to the instance selector. The finally model obtains a better classifier and high-quality data set (Feng et al., 2018). A deep reinforcement learning framework for time series data classification is established. This framework use a specific reward function and a clearly formulated Markov process (Martinez et al., 2018). The information available about imbalanced data classification with reinforcement learning is quite limited.

3 TECHNICAL BACKGROUND

3.1 Q-Learning

Q-Learning is one far-reaching reinforcement learning techniques that does not require a model of the environment to learn to execute complex tasks. Essentially Q-Learning makes possible for an algorithm, to learn a sequential task, where rewards are released in a step by step fashion, until a journey called ”Episode” is completed. After training the ”educated” agent develops a road map memory called ”policy”, usually represented by a matrix Q, which optimizes rewards capture trajectories in any definable environment. $Q(s_t, a_t)$ gives the value of taking action a_t in a state s_t. Equation 1 is the leading actor of the Q-learning algorithm, derived from the Bellman equation by considering the first and second term of an infinite series (Watkins and Dayan, 1992):

$$Q_{obs}(s_t, a_t) = r + \gamma max_a Q(s_{t+1}, a_{t+1}), \qquad (1)$$

where γ is the discount factor which manages the balance between immediate and future rewards. In this equation the value of $Q(s_t, a_t)$ of state and action is given by the sum of the reward r with the discounted maximum future expected reward after moving to the next state S_{t+1}. The value of $Q_{obs}(s_t, a_t)$ is computed by an agent and then is updated with its the own estimate of $Q^*(s_t, a_t)$ in a Q-table. The term $max_a Q(s_{t+1}, a_{t+1})$ gives the maximum value for all actions in the following state. Q-learning is an off-policy algorithm since during training it updates the Q-values without making any assumptions about the actual policy being followed (Li et al., 2019).

It should be noted that in low-dimensional finite state spaces, Q-functions are recorded by a table (matrix). However in high-dimensional continuous state spaces, Q functions cannot be resolved unless a deep Q-learning algorithm is used as mediator to fit the Q-function with a deep neural network producing feature rich data (Lin et al., 2019).

3.2 Data Description

The dataset contains transactions made by credit cards in September 2013 by European cardholders, it represents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions (MLG-Kaggle, 2015), notice the strong unbalancing between fraud and legal trades, typical of this kind of big business. The database contains numerical variables which have been hidden using PCA (Principal Component Analysis) transformation. These 28 features named $V1, V2, ..., V28$ contain confidential data of the users and are non-reversible, thus protecting the original characteristics of the data. There are two special features that have not been transformed using PCA, these are *Time* and *Amount*. There is also an important variable called *Class*, which is a fundamental value in the database.

The feature called **Amount** is the amount of money in each transaction. The largest transaction that has this data set is $25,691.16 while the average of the transactions is $88.35. Figure 1 shows that the data is mostly concentrated at very small values close to zero while only a few transactions approximate the maximum value found. On the other hand, the representation of the amount of money for each transaction (see Figure 2) shows some values that differ from the others. These are called *outliers* and in this case they are transactions in which a large amount of money is transferred. Logically, these values attract the attention of being possible frauds, however this is something that fraudsters want to totally avoid. Existing information shows that fraudsters frequently transferred small amounts of money to continue stealing in an undetectable manner.

The class feature is what gives information that if the transactions are fraudulent or not, this variable takes value 1 in case of fraud and 0 otherwise. This feature shows that there is a minimum percentage of fraudulent cases which represent 0.17% of all data. While non-fraudulent cases equal 99.83%. It is concluded that the data is highly imbalanced, which requires choosing appropriate measures to divide the data and make the training of the system effective.

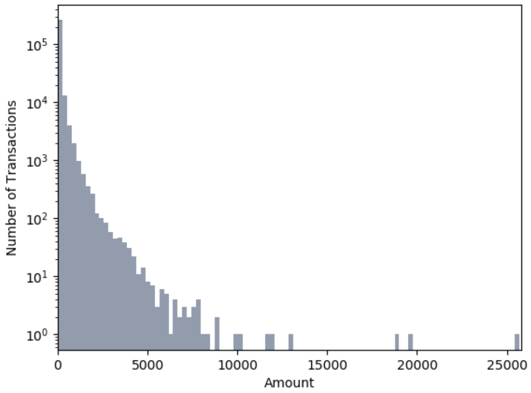


Figure 1: Distribution of Monetary Value Feature

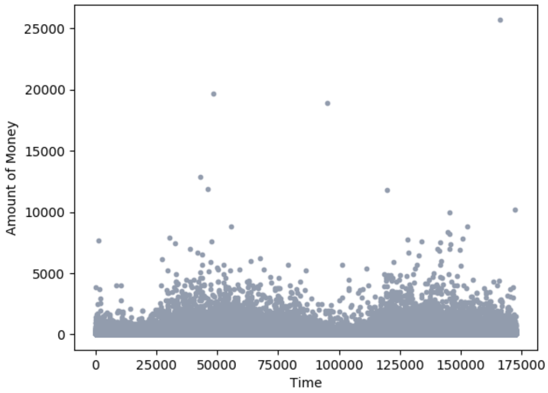


Figure 2: Money per Transaction

4 PROPOSED SOLUTION

This Section describes the proposed intelligent system, which learns to detect fraudulent transactions efficiently. The **Q-Credit Card Fraud Detector** (Q-CCFD) is composed of three different artificial intelligence techniques related to each other.

The first is an Auto-encoder (AE), which allows to extract important features from the unlabeled data. The hidden layer of this unsupervised network is the input to another Mediator Network (MN) designed to classify each input with its respective label genuine/fraud. The intermediate layer of the Mediator becomes the input to an Agent which is responsible for the final real time classification of the transaction being processed.

This Section is divided in two sections: Section 4.1 presents all the design and architecture of this proposed system and Section 4.2 which describes algorithm specifications used by these 3 artificial intelligence techniques.

4.1 Network Architectures

The system is the result of learning three main components which were adjusted individually. Figure 3 presents a general scheme of Q-CCFD in order to have a complete overview and simplify the definition of each component.

To set hyper-parameters to a convenient point is a common problem in neural networks, usually there exist a large number of possible combinations of which only few give the best results. Finding this optimal configuration requires effort but it is a unavoidable task. For this work, after some trial and error the following hyper-parameters are chosen.

4.1.1 Auto-encoder

This first component extracts important characteristics from the database, which is essential to facilitate the learning of the system and the detection of significant transactions that could harm the entire process.

Normalization The Auto-encoder inputs are the 28 features $V1, V2, ..., V28$ of the database. This data must be re-scaled so that they have a standard normal distribution with a mean of 0 and standard deviation of 1, this is compatible with sigmoid used as transfer function. As usual normalizing carried out before starting to train a model.

Before each X transaction is given as input of the AE, the maximum and minimum value of the 28 features is found and then the following equation is used:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}, \quad (2)$$

where each X transaction is rescaled at the interval $[0,1]$.

Number of hidden neurons and layers An Auto-encoder reproduces in its output the same data received in its inputs and has therefore the same number of neurons in both layers. On the other hand, the number of neurons in the hidden layer is a free choice element and it generally depends on the complexity of the relations in the database. The number of neurons per layers should be carefully chosen since too many or too few neurons will cause underfitting/overfitting problems.

We made comparative tests to determine the best number of hidden neurons for reliable representations; however, the definitive metric is determined based on the performance of the next operative network that is fed by the Auto-encoder. Finally, an AE with 38 neurons in a single hidden layer is chosen because with this numbers the output presents a minimum error value.

4.1.2 Mediator Network

A neural network is used to classify transactions. The input is given by the hidden layer of the Auto-encoder, therefore it does not require data normalization.

The output layer is given by 2 neurons, which correspond to the 2 possible cases: fraud or genuine. This is the base for a metric to measure the performance and find the best structure of the hidden layer, which will become important by defining the input for the next component.

After performing some tests which include passing information to other downstream components the best found possible structure was a hidden layer with 11 neurons.

4.1.3 Agent

The final component of the entire system is an agent who is responsible for accurately classifying in real time each one of the incoming transactions.

The agent input is the hidden layer of the MN. The output layer are 2 neurons corresponding to the two classes of the database. After some accuracy evaluation the amount of neurons in this hidden layer was set to 11.

4.2 Parameter Setting

This Section details system specifications such as the learning algorithm, the transfer functions, loss functions and other fundamental characteristics tuned to obtain good performance.

4.2.1 Auto-encoder

An auto-encoder uses an unsupervised learning. *Backpropagation algorithm* while setting the target values to be equal to the inputs. The transfer function that computes the weighted sum of inputs is *sigmoid function* and the loss function is *Mean Squared Error*. The best found learning rate value is 0.9.

4.2.2 Mediator Network

This supervised learning network uses *Backpropagation algorithm* with *MSE* and *learning rate* equal to

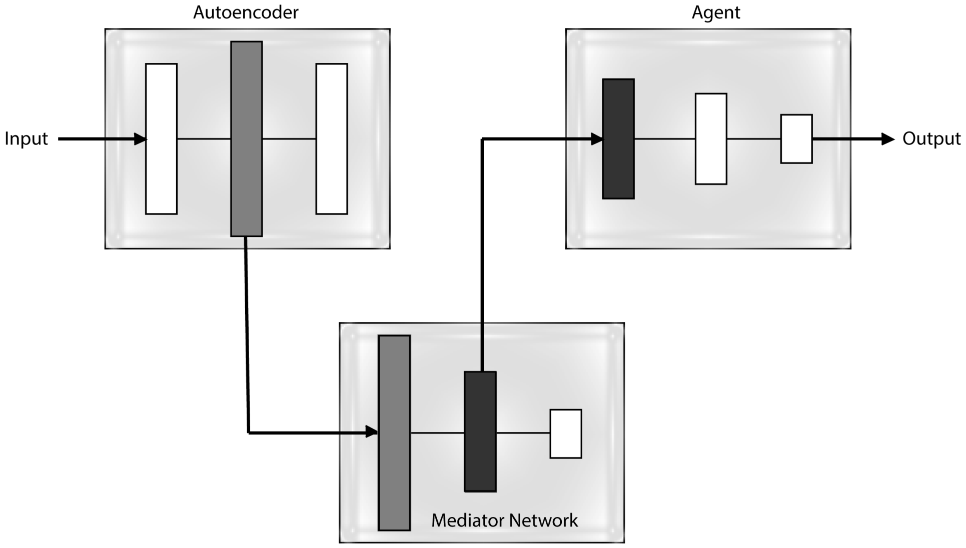


Figure 3: General Architecture of Q-Credit Card Fraud Detector

0.9. *Sigmoid* is the activation function that takes the input and compresses the output in the interval $[0.1]$.

4.2.3 Agent

The utilized Agent is based in reinforcement learning. Its activation function is the learning *sigmoid* and the algorithm is a novel version of Deep Q-learning (Mnih et al., 2015). Concerning framework our method solves classification problems using Reinforcement learning and an agent which receives rewards depending on the class to which it classifies incoming data. A positive reward is given it classifies a transaction correctly, otherwise a negative reward is assigned.

Imbalanced Classification Markov Decision Process (Lin et al., 2019) decomposes the task of classification into a sequential decision-making problem. The input received by the agent is each of the weights x_i of the hidden layer of the neural network with its respective label l_i. Therefore the training data set is $D = (x_1, l_1), (x_2, l_2), ..., (x_i, l_i)$. The ICMDP uses a five-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{E})$ with the following definitions:

- State $\mathcal{S}$: Each state is defined by the input, with x_1 being the initial state s_1. It goes to the next state with the next transaction given by hidden layer of NN.
- Action $\mathcal{A}$: The action a_t that the agent will take is given by the dataset label. In database used, the action is given by $A = \{0, 1\}$ being 1 for cases of fraud and 0 for non-fraud.
- Transition probability $\mathcal{T}$: The agent moves from one state s_t to the next state s_{t+1} according to the order in which the data is read. Transition probability is defined by $p(s_{t+1}|s_t, a_t)$.
- Reward $\mathcal{R}$: the value of the reward r_t depends on the action the agent takes. Two different rewards are defined for each class. The reward function is given by:

$$R(s_t, a_t, l_t) = \begin{cases} +1 & a_t = l_t \text{ and } s_t \in D_F \\ -1 & a_t \neq l_t \text{ and } s_t \in D_F \\ +\lambda & a_t = l_t \text{ and } s_t \in D_N \\ -\lambda & a_t \neq l_t \text{ and } s_t \in D_N \end{cases}$$

  where D_F represents the set of fraudulent transactions, D_N is the set of non-fraudulent transactions and λ is a value in the interval $[0, 1]$ that is the reward when the agent correctly classifies a non-fraudulent transaction, which are part of the class with the largest samples of the entire imbalanced dataset. The reward of $1/-1$ is greater than λ since being the minority class the impact of correctly classifying significantly affects the performance of the system.
- Terminal $\mathcal{E}$: Set to indicate that training must be stopped by the appearance of some terminal state. These states can arise in any training episode.

Episode is a concept in reinforcement learning, which is a transition trajectory from initial state to terminal state $\{s_1, a_1, r_1, s_2, a_2, r_2, ..., s_t, a_t, r_t\}$. Episode ends when all training data is classified or when the agent misclassifies the sample from minority class (Lin et al., 2019) and then the *terminal* variable changes its value, to start a new episode.

- Policy π_θ: This is a mapping function $\pi : \mathcal{S} \rightarrow \mathcal{A}$ where π_θ denotes the action a_t performed by agent in state s_t. The policy π_θ in ICMDP can be considered as a classifier with the θ parameter (Lin et al., 2019).

The above-mentioned definitions are an essential part of formally raising the problem and seeking to optimize the classification policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ which maximized rewards in ICMDP.

The final value with the best accuracy value of λ is 0.1. The memory replay size M used is 200000 and γ value is equal to 0.9.

The above-mentioned definitions are an essential part of formally raising the problem and seeking to optimize the classification policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ which maximized rewards in ICMDP.

The final value with the best accuracy value of λ is equal to 0.1. The memory replay size M used is 200000 and γ value is equal to 0.9.

5 IMPLEMENTATION

This Section describes the implementation of *Q-Credit Card Fraud Detector*, which makes possible to tune various parameters and then evaluate the performance of the overall system. Section 5.1 specifies the programming language and libraries used and Section 5.2 specifies the process of implementation.

5.1 Technology

The language used is C/C++ because it is flexible and implemented code is easy to read, understand and edit.

The working software runs in a C++ compiler with minimum graphic capacities and IDE, including our own neural and Q-Learning libraries.

5.2 Code Structure

Algorithm 1 shows an implementation of the general scheme that presents the Figure 3, in which each of the components of the proposed system is described, so that the three main techniques of artificial intelligence used can be distinguished.

```
Algorithm 1: Q-CCFD
  Input : $T = \{1,2,3,...,284807\}$
          $x_i = \{v1, v2, ...v28\}$, with $i \in T$
          Read Training Data
          Normalize $x_i \in D$
          $D = \{(x_1,..,l_1),(x_2,l_2),...,(x_T,L_T)\}$
  Randomly initialize weights $\theta_{IHO}$
  while Key not pressed do
      $\theta_{AE}$=Train Auto-encoder(D)
      $\theta_{MN}$=Train Mediator Network($\theta_{AE}$)
      Train Agent($\theta_{MN}$)
  end
  Testing: Data unlabeled
```

The function *Train Auto-encoder (D)* receives the normalized database as an input parameter and begins modifying its weights of the hidden layer θ_{AE} until the best representation of its input is obtained in its output. While *Train Mediator Network (θ_{AE})* starts to train a MN using as input the θ_{AE} values. Set the targets using the label of each transaction $\{0, 1\}$.

Agent learning uses *Train Agent(θ_{MN})* the algorithm 2, this is an implementation base on *Deep Q-learning for Imbalanced Classification Markov Decision Process* (Lin et al., 2019).

In each iteration, the agent uses policy $a_t = \pi_\theta(s_t)$ to choose the action and then obtains the reward using REWARD function in Algorithm 3.

6 RESULTS

Figure 4 shows the proposed system interface, a friendly and simple software that allows the user to have an easy interaction. Table 1 show a general summary of the Sections 4.1 and 4.2, which defines the best parameter settings for the system.

Table 2 shows the best result, which is given by the *accuracy* of the system. These values were measured in training and testing phase using whole imbalanced database. There are other algorithms that use the same dataset to detect fraud. The difference with Q-CCFD is that they use methods to balance the database: hybrid sampling and random sampling (Dighe et al., 2018). Q-CCFD when using the imbalanced dataset is more adaptable to the real problems that arise in bank transactions. Table 3 shows a comparison of the accuracy values obtained between these algorithms.

Algorithm 2: Train Agent

```
Input : $\theta_{MN}$

$D = \{(\theta^1_{MN}, l_1), (\theta^2_{MN}, l_2), ..., (\theta^T_{MN}, L_T)\}$
        Episode number K.
Initialize replay memory $D$ with $M$ capacity
Randomly initialize parameters $\theta$
Initialize simulation environment $\varepsilon$
for $episode\ k = 1\ to\ \boldsymbol{K}$ do
    Shuffle $D$
    Initialize state $s_1 = \theta^1_{MN}$
    for $t = 1\ to\ T$ do
        Pick an action: $a_t = \pi_\theta(s_t)$
        $r_t, terminal_t = REWARD(a_t, l_t)$
        Set $s_{t+1} = \theta^{t+1}_{MN}$
        Store $(s_t, a_t, r_t, s_{t+1}, terminal_t)$ to $M$
        Randomly sample
          $(s_j, a_j, r_j, s_{j+1}, terminal_j)$ from $M$
        Set $y_j =$
          $\begin{cases} r_j, & terminal_j = 1 \\ r_j + \gamma max_{a'} Q(s_{j+1}, a'; \theta), & terminal_j = 0 \end{cases}$
        Perform a gradient descent step:
        $L(\theta) = (y_j - Q(s_j, a_j; \theta))^2$
        if $terminal_t = 1$ then
            break
        end
    end
end
```

Algorithm 3: Environment Simulation

```
$D_F$ represents the fraud class
$D_N$ represents the non-fraud class
Function: $REWARD(a_t \in \mathcal{A}, l_t \in L)$
    Initialize $terminal_t = 0$
    if $s_t \in D_F$ then
        if $a_t = l_t$ then
            Set $r_t = 1$
        else
            Set $r_t = -1$
            $terminal_t = 1$
        end
else
    if $a_t = l_t$ then
        Set $r_t = \lambda$
    else
        Set $r_t = -\lambda$
    end
end
return $r_t, terminal_t$
```

	Accuracy	
	Training	Testing
Auto-encoder	99.9	96.4
Mediator Network	98.2	96.7
Agent	98.9	98.1

Table 2: Results of Q-Credit Card Fraud Detector.

Classifiers	Accuracy
Logistic Regression	96.2*
Naive Bayes	96.9*
Decision Tree	96.4*
K-Nearest Neighbour	99.1*
Q-Credit Card Fraud Detector	98.1

Table 3: Comparison with other existing algorithms. *Balanced dataset.

7 CONCLUSIONS

This paper presents a credit card fraud detector based on deep networks and reinforcement learning. It uses an auto-encoder and neural agent trained with a Q-learning algorithm working on imbalanced data set, treated by PCA and containing real credit card transactions. The system performs credit cards transaction classification by combining supervised, unsupervised and reinforcement learning. The proposed solution works as a quick acting intelligent agents and can identify frauds with remarkably accuracy.

The future scope is to adjust our solution to work in real-time banking systems. For this, a more complex database should be obtained (e.g., credit card holder id, where the transaction was realized, salesman id) and improve the solution if necessary.

ACKNOWLEDGEMENTS

Authors thank to the SDAS Research Group (`www.sdas-group.com`) for its valuable support.

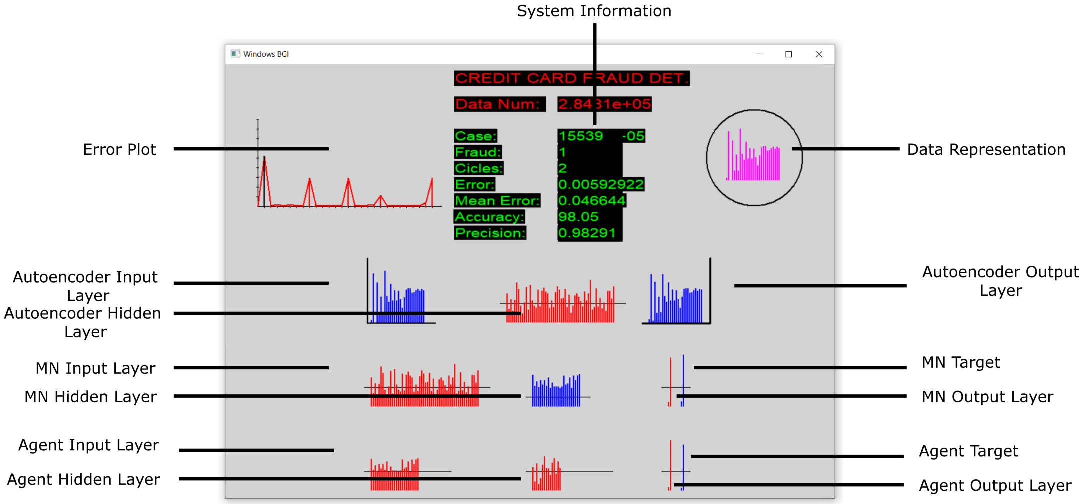
REFERENCES

Chukwuneke, C. (2018). Credit card fraud detection system using intelligent agents and enhanced security features.

Dighe, D., Patil, S., and Kokate, S. (2018). Detection of credit card fraud transactions using machine learning algorithms and neural networks: A comparative study. pages 1–6.

	Auto-encoder	Mediator Network	Agent
Input Layer	28	56	25
Hidden Layer	56	25	15
Output Layer	28	2	2
Learning Algorithm	Backpropagation	Backpropagation	Deep Q-learning
Transfer Function	Sigmoid	Sigmoid	Sigmoid
Loss Function	MSE	MSE	$(y_j - Q(s_j, a_j; \theta))^2$
Learning Rate	0.9	0.9	0.9
Discount factor (γ)	N/A	N/A	0.9
Lambda value (λ)	N/A	N/A	0.1
Size replay memory (M)	N/A	N/A	200000

Table 1: General specifications of Q-Credit Card Fraud Detector.



```
CREDIT CARD FRAUD DET.
Data Num:    2.8431e+05
Case:        15539   -05
Fraud:       1
Cicles:      2
Error:       0.00592922
Mean Error:  0.046644
Accuracy:    98.05
Precision:   0.98291
```

Figure 4: Interface of Q-Credit Card Fraud Detector

Dong, Q., Gong, S., and Zhu, X. (2019). Imbalanced deep learning by minority class incremental rectification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(6):1367–1381.

Feng, J., Huang, M., Zhao, L., Yang, Y., and Zhu, X. (2018). Reinforcement learning for relation classification from noisy data.

Li, M.-J., Li, A.-H., Huang, Y.-J., and Chu, S.-I. (2019). Implementation of deep reinforcement learning. In *Proceedings of the 2019 2Nd International Conference on Information Science and Systems*, ICISS 2019, pages 232–236, New York, NY, USA. ACM.

Lin, E., Chen, Q., and Qi, X. (2019). Deep reinforcement learning for imbalanced classification. *CoRR*, abs/1901.01379.

Martinez, C., Perrin, G., Ramasso, E., and Rombaut, M. (2018). A deep reinforcement learning approach for early classification of time series. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2030–2034.

MLG-Kaggle (2015). Credit card fraud detection dataset. Retrieved June 5, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Pumsirirat, A. and Yan, L. (2018). Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. *International Journal of Advanced Computer Science and Applications*, 9.

Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., and Kennedy, P. (2016). Training deep neural networks on imbalanced data sets. pages 4368–4374.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Wiering, M., Van Hasselt, H., Pietersma, A.-D., and Schomaker, L. (2011). Reinforcement learning algorithms for solving classification problems. pages 91 – 96.

Wiese, B. and Omlin, C. (2009). *Credit Card Transactions, Fraud Detection, and Machine Learning: Modelling Time with LSTM Recurrent Neural Networks*, volume 247, pages 231–268.