2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 11-14, 2020. Toronto, Canada

Incremental Neural-Network Learning for Big Fraud Data

Farzana Anowar
Department of Computer Science
University of Regina
Regina, Canada
anowar@uregina.ca

Samira Sadaoui
Department of Computer Science
University of Regina
Regina, Canada
sadaouis@uregina.ca

***Abstract*—Fraud detection systems aim to process a massive amount of data at high speed. To address the issues of data scalability, we introduce a chunk-based incremental classification approach based on a neural network (MLP) and a memory model to tackle the stability-plasticity dilemma. The incremental approach adapts the fraud model sequentially with incoming data chunks and retains past chunks a little more. We employ a large-scale credit-card fraud dataset that we organize into initial and incremental chunks for training and testing. Using data sampling, we solve the data skew problem, a critical issue in fraud detection. After each incremental phase, we evaluate the performance of the adjusted MLP classifier using the testing chunk. The experimental results demonstrate the effectiveness and efficiency of our incremental method and its superiority to the non-incremental MLP.**

***Index Terms*—Incremental Classification, Neural Networks, Big Data, Highly Imbalanced Data, Credit-card Fraud Data, Sliding Window.**

I. Introduction

Nowadays, a good number of real-world applications are subject to fraud, such as credit cards, telecommunication, e-auctions, insurance, and e-commerce. Detecting fraud is considered a difficult problem to solve due to three main reasons: (1) the amount of data to be processed and analyzed is massive, (2) data are generated continuously and in high speed, and (3) fraud activities must be detected in real-time even though under limited CPU and storage resources. Regular machine learning algorithms cannot handle the high data volume nor the rapid data generation. To address these issues, in this study, we develop an efficient incremental learning approach that is capable of functioning in real-world fraud detection scenarios. Here, incremental learning refers to the online classification of data that are presented sequentially in the form of mini-batches (called here chunks). Incremental learning aims to improve the existing model's knowledge but without retraining the model from scratch and without discarding the previously learned knowledge [1], [2]. Nevertheless, labeling fraud training datasets is a laborious task [3]. Our previous work [4] addressed the issue of unlabeled data with an incremental SGD classifier and validated the approach through a medium-sized auction fraud dataset. In contrast, in this present work, we develop an incremental neural network approach to deal specifically with big data and highly imbalanced data in the banking domain.

To tackle our problematic fraud learning task, we should select a classification algorithm that can be trained incrementally and that possesses high-speed execution when classifying unseen data. To this end, we select Neural Network (NN) algorithms, more precisely Multi-Layer Perceptrons (MLPs), because the latter supports the partial fitting mechanism so that data can be processed sequentially in the form of chunks. Also, MLPs are trained using back-propagation, so in this way, the weights and biases can be optimized much better to achieve the minimum error [5], which is crucial in the fraud detection domain. In this paper, we develop a chunk-based incremental MLP classification algorithm by selecting the appropriate architecture and configuration. Using the sliding window concept, we describe the memory model that is used to tackle the stability-plasticity dilemma in incremental learning [6]. On the one hand, we implement a stable model, i.e., the incremental MLP is capable of retaining outdated chunks a little longer, and on the other hand, MLP does not suffer from a loss in plasticity, i.e., the model is capable of processing new data rapidly. Surprisingly, the literature does not provide any explanation of the memory model for incremental learning. In this research, we describe the incremental memory model adequately.

We assess the effectiveness and efficacy of the proposed adaptive approach using a large-scale credit-card fraud dataset. To be able to perform the incremental classification, we split the fraud dataset into initial and incremental data chunks for training and testing phases. However, the training chunks are highly imbalanced. In this situation, the fraud class will be poorly misrepresented by the classifier, which is a serious concern since the fraud class is the target for investigation [7]. To handle this concern, we adopt an over-sampling technique. We develop the initial MLP classifier and then incrementally train it with several training chunks, while concurrently assess the detection and misclassification rates of the each produced fraud model using the testing (unseen) chunks. In the incremental learning setting, the performance evaluation is more problematic as it is assessed at different times to check its improvement in the long run. Additionally, to further validate the proposed incremental MLP algorithm, we compare it with the non-incremental MLP. For this

978-1-7281-8526-2/20/$31.00 ©2020 IEEE

purpose, we develop several MLP models from scratch using the accumulated training data at each incremental step.

We organize the rest of the paper as follows. In Section II, we discuss recent works on incremental NNs. In Section III, we introduce our chunk-based incremental MLP approach and its underlying architecture, configuration and memory model. In Section IV, we prepare the large-scale highly imbalanced credit-card fraud dataset for the incremental classification task. In Section V, we develop the base fraud MLP classifier using an optimization method, train it with the initial training and then assess its performance on unseen data. In section VI, we evaluate the MLP classifier's performance after each incremental learning phase. In Section VII, we compare the self-adaptive incremental MLP with a stationary MLP approach. In Section 8, we summarize our findings.

II. Related Work

In [8], the authors developed a NN incrementally to classify textual data by combining multiple sub-nets. Each time when new data become available, a corresponding sub-net is built. Then, all the sub-nets are combined to produce a final NN that finds patterns from input attributes. The input and output nodes of the final NN are just the union of input and output nodes of the sub-nets. Based on training data, the NN adjusts the weights and structure by using an unsupervised learning method. When a sub-net is generated, it maps the relation between input and output nodes for the observed data and thus the combined NN can produce similar output for the same inputs, which eventually makes the learning process unsupervised. For the experiment purpose, the authors utilized Reuters-21578 dataset to demonstrate the effectiveness of this method and obtained the maximum F1-measure of 92.5%, which proves that the proposed method is able to classify text data efficiently.

The research in [9] presented a batch online learning algorithm for RBF networks based on self-organizing incremental NN (SOINN), and named it SOINN-RBF. The latter is implemented using the self-adaption technique of SOINN and the pruning policy of MRAN. It is also based on the blend of SOINN and least square method. The technique of SOINN is to maintain the NN's nodes and their connections. Upon arrival of a new input data, SOINN searches for two nearest nodes and assess the distances respectively with their radii. If the new data is within any of the node's radii, a connection is established between the two nodes and the nearest one is pulled towards the input data. Then, the radii of two nodes are adjusted to confirm that they cover all the associated neighbors. After adapting SOINN, SOINN-RBF network introduces strategies for pruning, such as eliminating isolated nodes and oldest connections to remove noise and inappropriate connections respectively. To obtain the batch SOINN-RBF, all the hidden nodes are learned using SOINN and then LS algorithm is used to train the linear regression parameters. Based on both real-world and artificial datasets, the authors compared the performance of the proposed approach with GGAP-RBF, MRAN and SVM and obtained a comparable accuracy.

[10] introduces Incremental NN for Classification and Clustering (INNCC). This approach does not enforce any restriction on the NN structure. It is updated continuously using "competitive learning". INNCC is built by following three criteria: 1) an additional node is introduced only when required, 2) all the nodes from low-density spaces are removed, and 3) stability is ensured by learning new data. However, the removal of nodes should be under control. That is why the authors associated each node with a state of activated or inactivated. When INNCC training is complete, all the used nodes become inactivated and when new data is available for training, the nodes become activated. In this way, the ineffectual nodes are eliminated more significantly. During training, both unlabeled and labeled data are utilized to learn the structure of the underlying data space. First, labeled data is used to train a classifier and after that, unlabeled data that are classified with high level of confidence are included incrementally in the labeled dataset with the use of their predicted labels. SVM is utilized to verify the labels and consider the the trusted data only. This procedure keeps repeating until all the unlabeled data are annotated. Two synthetic and four real-world datasets have been used for validating the proposed framework, which returned a promising performance.

In the context of data stream with concept drift, the authors in [11] proposed an incremental NN with a parameter optimization module for finding the optimal parameters' values so that the network can achieve the maximum performance. To fix the issue of the heavy time consumption of the optimization method, they proposed a loss function that has the ability to skip the optimization when necessary (in case of concept drift). Also, to make the NN incremental, they used the sliding window concept where they divided the dataset into 20 portions with the same size. Since the network structure is not changing during the training phase in the proposed method, hereby it is anticipated that the complexity of NN stays steady for each incremental step. During the incremental learning, the authors adopted a "learn-and-forget" method. While new data appears, some learned knowledge are going to be substituted by new-found knowledge from new data. However, in a NN, this occurs by updating the weights progressively. With this method, the NN with the simple structure holds short and long-term memory. Later, for choosing the optimal parameter values, the authors used CVParameterSelection from Weka for the learning rate and momentum. For the experiments, one real-world dataset has been used and the results showed that the proposed method returned a consistently increasing classification accuracy, unlike an incremental NN without any optimization for a larger window size.

The authors in [12] proposed a novel incremental NN for unsupervised learning named SOINN+. SOINN+ is basically different from SOINN (which learns from non-stationary data) on the "forgetting" mechanism. Previously in SOINN, the deletion of edges and nodes is decided by two parameters

that are optimized using cross-validation or similar techniques. SOINN+ alters this shortcoming by deleting the edges and nodes only if they are irrelevant for the learning process, which eventually ends up having better forgetting mechanism. To attain more graceful forgetting, the authors developed three new concepts for SOINN+, which are: 1) trustworthiness, 2) idle time, and 3) (un-)utility of a node. In the experiment, SOINN+ has been used to identify the clusters with five other stream clustering techniques in noisy data streams with the presence of recurring and sudden concept drifts. The experimental results with real-world and synthetic datasets revealed that SOINN+ is able to detect the real clusters and capable of maintaining the detected structures even under recurring and sudden concept drifts than other techniques.

Recently, the research [4] proposed a chunk-based incremental classifier to address the labeling problem of shill bidding data in commercial e-auctions. The authors selected the SGD classifier to develop the incremental approach because it is independent of the size of the training datasets, crucial in real-world auction scenarios. They showed how to adjust the SGD classifier gradually with the training chunks. According to the experiments, the performance of the sequential models was increasing for each chunk. In contrast, the misclassification rate, speed, and loss decreased steadily.

III. Chunk-based Incremental MLP

Incremental learning characterizes a dynamic approach that can be utilized in different situations, like when training data becomes available over time, or when the training dataset is large and out of memory limits. Generally speaking, an incremental approach aims to adapt a current classification model with new data, one by one, but without retraining the model from scratch and without discarding the previously learned knowledge [1], [2].

A. MLP Model

Among the classification algorithms, we choose the Multi-Layer Perceptron (MLP) for two significant reasons: (1) MLP is able to support the "partial fit" mechanism so that data can be processed sequentially in the form of mini-batches (called here chunks), and (2) MLP is trained using back-propagation, so in this way, all the weights and biases can be optimized efficiently to achieve the minimal error. MLP consists of several fully connected layers where each node is independent of the rest of the nodes of the same layer, which means each node has a unique set of weights and a bias. In addition, every node employs a nonlinear activation function, except the input nodes. Since we are dealing with a two-class learning problem, the output layer contains a single node to make the prediction based on the threshold of 0.5 to assign an instance with its label as shown below (v is the average weighted summation of all the input values of the output node):

$$label = \begin{cases} 0\ (Normal) & \text{when activationFunction(v)} < 0.5 \\ 1\ (Fraud) & \text{when activationFunction(v)} \geqslant 0.5 \end{cases}$$

MLP possesses three hyper-parameters to be tuned: learning rate, number of epochs and batch size. The learning rate denotes the step size on how much a weight will be updated. With one epoch, the whole dataset is passed both forward and backward. Batch size specifies the number of data that is going to be propagated at once through the network. It is suggested to have a small batch size as it requires less memory and therefore the whole training process becomes faster [13]. The classifier's loss in each epoch is measured with an objective error function, such as the cross-entropy, and minimized using for example a gradient descent-based optimizer.

To conduct the experimental investigation, we employ Scikit-learn from the Anaconda distribution so that training data are accepted and learned incrementally. MLP provides an early stopping criteria (only when the optimizer is SGD or Adam) to terminate training when the validation score (10% of training data) is not improving anymore. If there are k training data, n features, m hidden layers, each hidden layer contains h neurons, and z output neurons, then the time-complexity for back-propagation is $O(k.n.h^m.z.i)$, here i is the iterations number [14]. Since back-propagation comes with high execution time, it is advised to begin with lesser numbers of nodes and hidden layers.

B. Incremental Memory Model

The purpose of our chunk-based incremental classification approach is to learn with big data that would not fit in memory as a whole rather data are provided chunk by chunk. Nevertheless, a major challenge when adopting the incremental learning is the stability and plasticity issue. Indeed, any incremental model needs to be stable enough to hold training information a little longer, and at the same time, it needs to forget (plasticity) previous data gradually, otherwise the model will not be able to adapt with incoming data properly [15].

The ability to learn incrementally from a data chunk guarantees that at any given time, there will be a reasonable amount of data in the local memory [14]. The incremental MLP model does not immediately forgets past information of data chunks, instead these information remain in the memory little longer. This way the older chunks are forgotten gradually. In Fig. 1, we use the 'Sliding Window' concept to describe how incremental learning is handled by Scikit-learn. In our case, the window size is the size of each chunk, which is fixed. Since, in NN, the gradient of the loss function is updated with the most recent data, hence, newer chunks have more influence on the state of the MLP model than older chunks. As a result, recent chunks are more vivid in the model's memory. Moreover, an optimizer involves a large amount of memory to undertake the expensive optimization operation, and this will exhaust the memory. To fix this issue, Scikit-learn offers users by default 1 GB of local working memory. Hence, computations can be performed inside a large amount of memory. Furthermore, the tool provides us with two libraries, "sklearn.set_config" and "config_context", to allow users to reduce or increase the memory size if necessary. In case of a huge memory

requirement, we can also utilize the cloud service provided by Anaconda to store the fraud classification model and training data in the primary memory of Anaconda distribution.

C. Incremental Classification Algorithm

Before conducting any incremental adaptation, we first need to convert all the data chunks into linear data because real-world data are mostly non-linearly separable. For this purpose, several Kernel Map Approximation (KMA) functions have been defined, such as RBF, Polynomial, Sigmoid and Gaussian. We select RBF because it performed better for other fraud detection datasets [4]. Algorithm 1 presents the steps to perform a chunk-based incremental classification based on the meta-estimator "incremental" and the incremental memory model provided by Sckit-learn.

Algorithm 1 Chunk-based Incremental MLP

Inputs: initial training chunk, incremental training chunks initial testing chunk, incremental testing chunks
Output: adjusted and validated MLP

//Incremental API
1: Import "Incremental" meta-estimator to connect with "PartialFit"

//Data Preprocessing
2: Re-balance all chunks using data sampling
3: Convert all chunks to linear data using KMA

//Base Classification Model
4: Wrap MLP with Incremental meta-estimator
5: Build MLP architecture (number of features, number of layers and nodes, activation functions)
6: Configure MLP (optimizer, learning rate, loss function)
7: Train MLP with initial chunk
8: Evaluate MLP performance with initial testing chunk

//Incremental Classification Model
8: Repeat until last incremental chunk:
{
    8.1: Adapt MLP with incremental training chunk using incremental model memory
    8.2: Evaluate MLP performance with incremental testing chunk
}

IV. Credit-Card Fraud Data Preparation

To assess the effectiveness of our proposed approach, we employ a large-scale Credit Card Fraud Dataset (CCFD) consisting of 284,807 instances, made public in the Kaggle repository [16]. CCFD contains European cardholders' transactions within two days of the month of September 2013 and labeled as Fraud or Normal. The dataset possesses 30 numerical training features obtained with the feature extraction method PCA, but the two attributes 'Time' and 'Amount' have not been transformed. CCFD is highly imbalanced because the fraudulent transactions account only 0.172% of the whole dataset.

TABLE I: Credit-Card Fraud Data Splitting

Subset	Total	Training Chunk		Test Chunk	
		N	F	N	F
Init. Subset	124807	87183	181	37365	78
Inc. Subset#1	40000	27931	69	11970	30
Inc. Subset#2	40000	27975	25	11990	10
Inc. Subset#3	40000	27962	38	11984	16
Inc. Subset#4	40000	27968	32	11987	13

A. Splitting into Training and Testing Chunks

To be able to conduct the incremental learning task, we first divide CCFD into five subsets; the first subset should be sufficiently representative to develop properly the initial fraud classifier. We manually make the remaining subsets equal with 40,000 instances each. Based on the stratified splitting method, we further divide each subset to the ratio of 70:30 as training and testing (unseen) chunks. The splitting results are exposed in Table I where N denotes normal instances and F fraud instances.

B. Highly Imbalanced Training Chunks

As shown in Table II, all the training chunks have highly imbalance ratios; for example, Chunk#2 has an imbalance ratio of Normal to Fraud data equals to 1119:1, which is quite high. It is a well-known fact that in the situation of skewed class distribution of training data, the fraud class will be poorly represented since the classifier will be biased towards the Normal class [7], [17]. The fraud class is the most significant output since it is the target for investigation. Hence, we solve this serious issue by adopting data sampling techniques, such as oversampling with SMOTE. We try several imbalance ratios, from 3:1 to 10:1. The ratio of 3:1 returned very satisfactory performance unlike the other ratios because the number of fraud instances was still very low. Table II presents the imbalance ratios of the original training chunks and the corresponding re-balanced chunks. Since it is recommended that the testing data should have the same class distribution than the training data, we also re-sample all the testing chunks with the imbalance ratio of 3:1 (Table III).

TABLE II: Original Ratio and Re-Balanced Training Chunks

Training Chunk	Original Imbalanced Ratio (N:F)	Balanced Data	
		N	F
Init. Chunk	481:1	87183	26154
Inc. Chunk#1	405:1	27931	8379
Inc. Chunk#2	1119:1	27975	8392
Inc. Chunk#3	735:1	27962	8388
Inc. Chunk#4	874:1	27968	8390

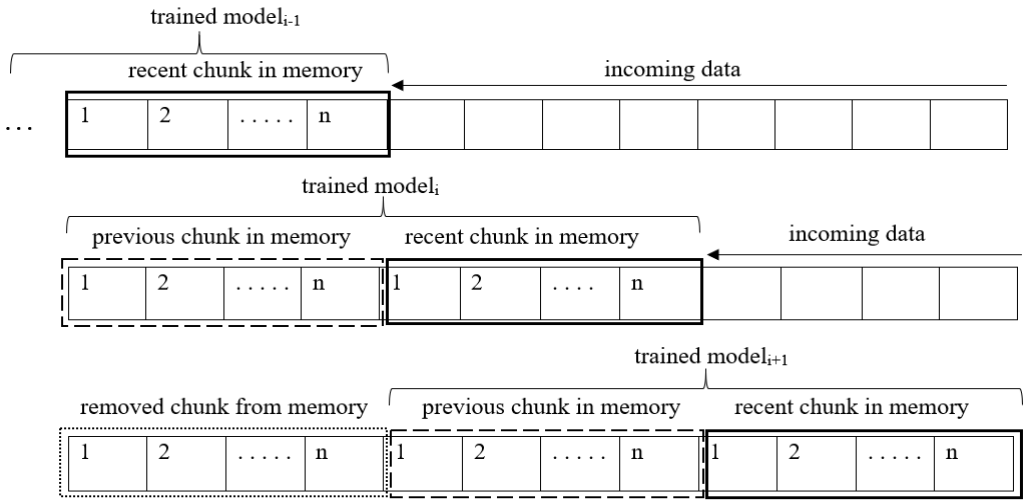

Fig. 1: Chunk-based Incremental Memory Model

TABLE III: Original and Re-Balanced Testing Chunks

Testing Chunk	Original Data		Balanced Data	
	N	F	N	F
Init. Chunk	37365	78	37365	11209
Inc. Chunk#1	11970	30	11970	3591
Inc. Chunk#2	11990	10	11990	3597
Inc. Chunk#3	11984	16	11984	3595
Inc. Chunk#4	11987	13	11987	3596

V. Base Fraud Classifier Development

Firstly, we develop a fully connected MLP with one hidden layer of 500 nodes. As the activation function, we adopt "ReLu" since it does not suffer from the vanishing gradient problem, accelerates the convergence and is less computationally expensive. All the weights/biases are randomly initialized using the activation function Relu of the hidden layer [14]. In Scikit-learn, only SGD and Adam optimizers allow an incremental classifier to receive data in the chunk form. We choose Adam because it converges faster when it is minimizing the loss function. MLP classifier utilizes by default the binary cross entropy (the log-loss) as the loss function for any two-class dataset. During back-propagation, we use momentum for speeding up the training process, and its value has to be between 0 and 1 when using Adam optimizer. We use the default value of 0.9 for momentum because a large value implies that the convergence will be faster for each epoch. In contrast, a smaller momentum can slow down the training process for a network. Now, if we have a large momentum, we then need to choose a lower value for the learning rate. If both momentum and learning rate have large values, then there is a chance of missing the global minima [18]. Since we have a high momentum, our parameter optimization method (explained below) will select a lower value for the learning rate as shown in Table IV. Lastly, to avoid overfitting, we set the regularizer term "alpha" to the default value of 0.0001.

We first train our chunk-based incremental MLP algorithm (see Algorithm 1) using the initial balanced training chunk exposed in Table I. Actually, we combine Randomized Search with 10-fold Cross-Validation to determine the optimal values for the MLP hyper-parameters, batch size, epochs and learning rate; Table IV exposes their candidates and optimal values. Moreover, we also check overfitting by extracting 30% from the first training chunk as the validation set. If the validation performance is worse than the training performance, then the model is overfitting. Based on the loss metric, Fig. 2 clearly depicts that when the number of epochs is increasing, the validation loss is decreasing more than the training loss, which proves that the initial MLP model is not overfitting.

Second, we apply the learned MLP model to the first balanced testing chunk presented in Table III. We report a high performance on the first testing chunk with a Recall of 91%, F-score of 90%, False Negative Rate (FNR) of only 9%, and an error rate of 2.2%. We choose FNR (i.e. the misclassified fraudulent transactions) because we are more focused on the fraud class.

It is worth mentioning that with the RBF function, we obtain better loss value than with the original first chunk. Indeed, we obtain the loss of 0.19 for the nonlinear initial chunk, but after applying the kernel function, we got a loss of 0.022 and this is due to the reduced space complexity. This statement holds true for the other data chunks.

TABLE IV: Parameter Tuning for Initial MLP Classifier

Hyper-parameter	Possible Values	Optimal Value
Batch Size	32, 64, 128, 256, 512 and 1000	128
Epoch	[10 - 500]	37
Learning Rate	[0 - 1]	0.0090506381

TABLE V: Predicting Performance of Static MLP Models

Training Data	Learned Model	Testing Chunk	Recall	FNR
Initial Chunk = 113337	MLP#1	48574	0.89	0.11
Previous Data+Chunk#1 (113337+36310) = 149647	MLP#2	15561	0.87	0.13
Previous Data+Chunk#2 (149647+36367) =186014	MLP#3	15587	0.885	0.115
Previous Data+Chunk#3 (186014+36350) =222364	MLP#4	15579	0.87	0.13
Previous Data+Chunk#4 (222364+36358) =258722	MLP#5	15583	0.915	0.085

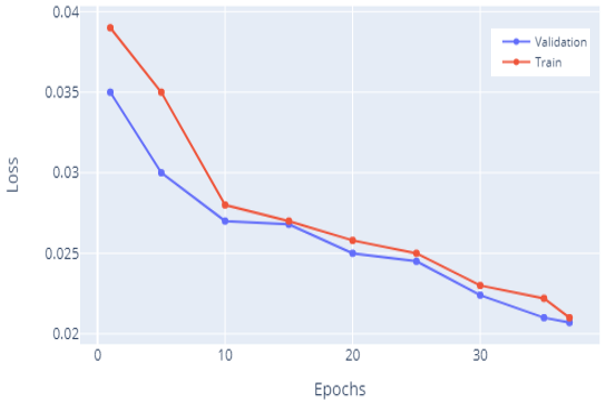

Fig. 2: Training and Validation Sets for Base MLP

VI. Incremental Classification

We also conduct Randomized Search combined with 10-fold CV when adjusting the MLP model with each incremental training chunk. Table VI presents the best values of the tuned hyper-parameters. We also verify that the fraud classifier does not overfit with each incremental training chunk. After the learning phase, as presented in Table VII, we compute the accuracy results for each incremental testing chunk (balanced and linearly separable). As we adapt the fraud classifier with more data, most of the time, the accuracy rate increases, and the misclassification error FNR and loss rates decrease.

TABLE VI: Parameters for Incremental MLP Models

Incremental Chunk	Batch Size	Epoch	Learning Rate
#1	512	56	0.0159041210
#2	128	60	0.0053328074
#3	32	60	0.0010923843
#4	128	75	0.0033198990

TABLE VII: Performance of Incremental MLP Models

Testing Chunk	Recall	F-score	FNR	Loss
#1	0.92	0.935	0.08	0.019
#2	0.94	0.94	0.06	0.015
#3	0.96	0.96	0.04	0.013
#4	0.96	0. 96	0.04	0.013

Furthermore, we employ three criteria, stability, improvement and recoverability, that have been mentioned in [15] to evaluate the robustness of any incremental classification approach:

- Stability means that the prediction performance should not vary drastically between the incremental adaptations. According to Table VII, it is evident that the testing accuracy increases with consistency with a 2% gap for Recall and between 1% to 2% gap for F-score.
- Improvement refers to the steady performance improvement when the incremental model is fed with more training data. From the first to the last incremental chunk, our fraud model returns a steady increment from 92% to 96% for Recall and from 93.5% to 96% for F-score.
- Recoverability means that the incremental model should recover from its errors. In our case, the misclassification error FNR decreases gradually as the fraud classifier receives more training data.

VII. A Comparison

In Table V, at each step, we compare the classification accuracy of the incremental MLP model with the non-incremental MLP models. Firstly, from scratch, we train at each step the static MLP algorithm using the accumulated training data (initial and incremental chunks). As a result, we obtain five different MLP models. We then compare the performance of incremental MLP with all the static MLPs. As we can see, incremental MLP is clearly superior to the static MLPs in terms of Recall and FNR, which is expected since incremental

learning has a better generalization capability than static learning as observed in the literature.

VIII. Conclusion

Banking systems manage an enormous amount of transactions daily. Hence, there is a significant need for an approach that can identify fraudulent activities accurately. For this purpose, we devised a chunk-based incremental MLP approach based on a memory model that implements the plasticity-stability mechanism so that the classifier can progressively learn from incoming transaction chunks and without catastrophic forgetting. Our approach also handles the highly imbalanced transaction chunks. Using a real credit-card fraud dataset, we first developed the initial incremental fraud MLP classifier, which returned a promising performance on unseen data chunks. Next, we conducted several incremental training phases, and the fraud classifier was able to increase its detection rate and lower its misclassification rate at each phase. Lastly, we showed the superiority of the incremental MLP classifier to the stationary MLP.

References

[1] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing, Elsevier*, vol. 275, no. 1, pp. 1261–1274, 2018.

[2] W. Zang, P. Zhang, C. Zhou, and L. Guo, "Comparative study between incremental and ensemble learning on data streams: Case study," *Journal Of Big Data, SpringerOpen*, vol. 1, no. 5, pp. 1–16, 2014.

[3] S. Elshaar and S. Sadaoui, "Semi-supervised classification of fraud data in commercial auctions," *Applied Artificial Intelligence*, vol. 34, no. 1, pp. 47–63, 2020.

[4] F. Anowar and S. Sadaoui, "Chunk-based incremental classification of fraud data," in *The Thirty-Third International Florida Artificial Intelligence Research Society Conference (FLAIRS)*. AAAI Press, 2020, pp. 176–180.

[5] H. Das, A. K. Jena, J. Nayak, B. Naik, and H. Behera, "A novel pso based back propagation learning-mlp (pso-bp-mlp) for classification," in *Computational Intelligence in Data Mining*. Springer, 2015, vol. 2, pp. 461–471.

[6] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *The 28th annual ACM symposium on applied computing*, 2013, pp. 801–806.

[7] F. Anowar and S. Sadaoui, "Detection of auction fraud in commercial sites," *Journal of Theoretical and Applied Electronic Commerce Research, Universidad de Talca*, vol. 15, no. 1, pp. 81–98, 2020.

[8] J. H. Wang and H. Y. Wang, "Incremental neural network construction for text classification," in *International Symposium on Computer, Consumer and Control*. IEEE, 2014, pp. 970–973.

[9] J. Lu, F. Shen, and J. Zhao, "Using self-organizing incremental neural network (soinn) for radial basis function networks," in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 2142–2148.

[10] A. Hebboul, F. Hachouf, and A. Boulemnadjel, "A new incremental neural network for simultaneous clustering and classification," *Neurocomputing*, vol. 169, pp. 89–99, 2015.

[11] S. Fong, C. Fang, N. Tian, R. Wong, and B. W. Yap, "Self-adaptive parameters optimization for incremental classification in big data using neural network," in *Big Data Applications and Use Cases*. Springer, 2016, pp. 175–196.

[12] C. Wiwatcharakoses and D. Berrar, "Soinn+, a self-organizing incremental neural network for unsupervised learning from noisy data streams," *Expert Systems with Applications*, vol. 143, pp. 1–33, 2020.

[13] J. Brownlee, "A gentle introduction to mini-batch gradient descent and how to configure batch size," https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/, 2019, last accessed 20 April 2020.

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

[15] X. Geng and K. Smith-Miles, *Incremental Learning*. Boston, MA: Springer US, 2009, pp. 731–735. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_304

[16] M. L. Group-ULB, "Credit card fraud detection," https://www.kaggle.com/mlg-ulb/creditcardfraud, 2018, last accessed 22 August 2020.

[17] F. Anowar, S. Sadaoui, and M. Mouhoub, "Auction fraud classification based on clustering and sampling techniques," in *The 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 366–371.

[18] D. McAuley, "The backpropagation network: Learning by example," http://staff.itee.uq.edu.au/janetw/cmc/chapters/BackProp/index2.html, 1999, last accessed 22 August 2020.