International Journal of Data Science and Analytics (2021) 12:165–174
https://doi.org/10.1007/s41060-021-00258-0

REGULAR PAPER

Incremental learning strategies for credit cards fraud detection

B. Lebichot[1] · G. M. Paldino[1] · W. Siblini[2] · L. He-Guelton[1] · F. Oblé[2] · G. Bontempi[1]

Received: 4 July 2020 / Accepted: 26 April 2021 / Published online: 9 June 2021

Abstract
Every second, thousands of credit or debit card transactions are processed in financial institutions. This extensive amount of data and its sequential nature make the problem of fraud detection particularly challenging. Most analytical strategies used in production are still based on batch learning, which is inadequate for two reasons: Models quickly become outdated and require sensitive data storage. The evolving nature of bank fraud enshrines the importance of having up-to-date models, and sensitive data retention makes companies vulnerable to infringements of the European General Data Protection Regulation. For these reasons, evaluating incremental learning strategies is recommended. This paper designs and evaluates incremental learning solutions for real-world fraud detection systems. The aim is to demonstrate the competitiveness of incremental learning over conventional batch approaches and, consequently, improve its accuracy employing ensemble learning, diversity and transfer learning. An experimental analysis is conducted on a full-scale case study including five months of e-commerce transactions and made available by our industry partner, Worldline.

Keywords Fraud detection · Incremental learning · Transfer Learning · Fintech

1 Introduction

Credit card fraud jeopardizes the trust of customers in e-commerce transactions. This led in recent years to major advances in the design of *fraud detection systems* (FDS) to detect fraudulent transactions within a short time and with high precision. Thanks to those improvements, the fraud loss is expected to decrease for the first time in decades [1]. Nevertheless, FDS need continual improvements to keep pace with fraudsters.

Streams of credit card transactions are processed by FDS by using both expert-based and data-driven detection models. Training a data-driven model is challenging because of *unbalancedness* [2] (frauds represent less than 1% of all transactions), *concept drift* (due to seasonal aspects and evolving fraudster strategies), *large size* (millions of transactions per day) and the *streaming nature* of the data. Because of the three first challenges, models actually used in production are often trained with batch learning [3], leading to suboptimal solutions in sequential settings. Batch learning indeed needs frequent retraining steps to cope with sequential data. However, a batch retraining with all available data is typically unfeasible because of technical, temporal and data storage limitations. For this reason *sliding window* mechanisms [4] are used as an intermediate solution by focusing the retraining on a fraction of recent samples.

Once the retraining rate coincides with the arrival rate, we are in an incremental learning setting. *Online learning* deals with sequential incoming data by returning a prediction after each single instance observation. Once the true class is made known (e.g., by an oracle), the model is updated in order to minimize a given *regret function*. Well-known online algorithms are online gradient descent [5] and online Newton step [6].

The efficiency of a learning approach in a sequential setting depends on the stationarity (or not) of the distribution underlying the observations. In the stationary case, the more the samples are considered, the more the accurate is the estimation of the underlying distribution. Also, stationary settings favor batch learning approaches for their convergence properties (to local or global optima). At the same time, those properties may be detrimental if we are confronted with nonstationarity or concept drift. It is well known that concept drift makes fraud detection a difficult task but the real chal-

✉ B. Lebichot
bertrand.lebichot@ulb.ac.be

[1] Machine Learning Group, Computer Science Departement, Faculty of Sciences, Université Libre de Bruxelles, Brussels, Belgium

[2] Development and Innovation, Worldline,Lyon, France

lenge remains how to find the appropriate trade-off between adaptation and learning or between forgetting and memory. In this paper, we propose to tackle concept drift using both the sliding window and the incremental strategy.

In [7], we introduced the use of an ensemble of classifiers trained on different windows of transactions to deal with concept drift. Ensemble methods are based on the idea that the more we combine decorrelated models the more we may reduce variance and ensure stability. The notion of diversity plays then a major role within an ensemble [8]. Moreover, a diversity-based ensemble allows safeguarding different concepts, ensuring both greater adaptability in case of recurring concepts and a reduced impact by negative historical models that would otherwise have a greater weight.

Another recent interesting approach for incremental learning has been proposed by [9]. DTEL (diversity- and transfer-based ensemble learning) combines *sliding window* [4], *transfer learning* [10] and *ensemble methods*. [11,12].

The DTEL procedure is the following: Each time a new chunk of data is available, a new decision tree classifier is trained from this chunk. Then the historical trees (built from previous data chunks) are transferred to fit the current chunk. (Two *transfer learning* mechanisms are allowed: Reset the class of the leaves and grow subtrees according to the current chunk.) The set of historical trees is constrained to a fixed size: The removed tree is the one leading to the largest diversity in the ensemble. (The Yules Q-statistic is used.) Finally, this ensemble votes, for each sample, with each tree being weighted by its own prediction error on the current chunk.

The rationale of transfer learning is to fill the gap between two supervised learning tasks by reusing what was learned from the former (called *source*) to better address the latter (referred to as *target*) [10]. In this sense, transfer learning may be useful to deal with concept drift if we assume that different concepts are somewhat related [10].

This paper aims to assess the interest of adopting incremental strategies to detect fraudulent credit card transactions in a real-world FDS setting. We aim to answer two main questions:

1. Is incremental learning competitive with conventional batch and retraining approaches?
2. Is a DTEL-like approach, combining incremental, ensemble and transfer learning robust with respect to concept drift?

In order to provide an answer, we designed and implemented 16 different approaches to deal with sequential streams of transactions. The comparative assessment is based on a five-month dataset (more than 50 million e-commerce transactions) provided by our industrial partner. For the sake of comparison, all the approaches rely on a dense neural net (NN), whose architecture had been defined by previous research. NNs are often used in fraud detection [13,14] and play a major role in continuous lifelong learning [15]. Also, there were a lots of recent work on fraud detection in the neural network community. It includes works on convolutional neural networks [16], autoencoder [17,18], long short-term memory and gated recurrent units GRU [19–21], resampling [19] and ensemble [21] of neural networks.

As far as the DTEL-like approach is concerned, we extend the DTEL [9] in three directions: (i) we move from decision trees to (GPU-trained) neural networks. This makes possible to test the approach in a more realistic and complex setting (original paper mostly includes shorter and synthetic streams); (ii) we assess several diversity criteria; and (iii) we adopt a transfer learning strategy designed on the purpose of fraud detection.

Note that our approach to deal with concept drift can be defined as *passive* according to the active/passive distinction made in [22]. Active adaptation, which requires a specific test to detect a distribution change [23], is suitable when a small number of major changes occur in time. This is not the case of fraud detection, where several drifts occur on a daily basis.

The rest of this paper is structured as follows: Sect. 2 introduces background and notation. Sections 3 and 4 analyze the two questions addressed in the paper, respectively. Section 5 proposes an additional assessment on a public dataset. Finally, Sect. 6 concludes the paper with some perspectives about the future.

2 Background and notation

Let us consider a FDS ingesting a stream of n credit card transactions $Tr_1, Tr_2, ..., Tr_n$. Streaming processing systems (e.g., Spark [24]) typically group them into *batches* (or chunks of data) according to specific criteria, like the maximum size of the batch or time interval (daily transactions).

Once the FDS raises an alert, the transaction is checked by investigators before deciding the relevant action (e.g., customer contact, block of the card, etc.) [7]. Given the limited number of investigators, it is crucial for a FDS to return the most accurate ranking (e.g., in terms of a conditional probability score) within the budget k of alerts that can be investigated (typically 100). Such accuracy is measured in terms of *precision* within the first k alerts, here denoted Pr@k. A measure of the accuracy of the entire ranking is the *area under the precision–recall curve* (AUPRC) known to be more suited for unbalanced classification than the area under the ROC curve [25,26]).

A peculiarity of real FDS is the delayed feedback [7], due to the fact that transaction labels are made available only several days later, once customers have reported unauthorized transactions. The delay in obtaining accurate labels and the

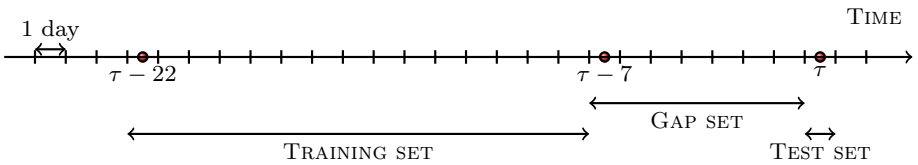

Fig. 1 Real-life FDS scenario with three sets of data. It takes several days to inspect all transactions, mainly because it is sometimes the card holder who reports undetected frauds. Hence, in practice, the fraud tags of the *gap set* are unknown. This scenario is repeated each day, with the sliding window approach, as the parameter τ is incremented

interaction between alerts and supervised information has to be carefully taken into consideration when learning in a concept-drifting environment: It is represented by the "*gap set*" (also simply called gap) of Fig. 1 for a batch learning perspective. The gap set is the part of the data which cannot be considered as annotated yet. In our experiment, in accordance with field experts, the gap is seven days long.

Notice that in Fig. 1, τ refers to the testing day (and therefore the whole *sliding window*) and that models are built on 15 previous days, taken after the gap set. By changing τ, we get different testing days. Some strategies to incorporate the gap set data can be found here [27,28]. In our case, gap set data are just discarded.

2.1 Data and code

The database is made up of about 50M e-commerce transactions occurred during 153 days (61 for model initialization, 7 gap days, 15 validation days, to set the hyperparameters, and 70 test days). The fraud ratio is 0.201%, and each transaction is described by 23 features. All data are standardized. Validation days are used to tune the hyperparameters introduced in the methodological section and detailed in Table 3. Though data cannot be made available for confidential reasons, consider that the data distribution is similar to the one of the Kaggle dataset [29], an older, two-day long, anonymized extract from the same database. Section 5 shows the replication of the main findings of this paper on this widely used dataset.

Note that several choices in our approach are due to the fact that, in data provided by the industrial partner, the time-related information is featurized into time-based aggregates [28]. This reduces the potential of techniques like time-based CNN [16] or LSTM, recently adopted to detect frauds [30] in sequence of transactions (e.g., at card holder level).

All experiments were carried on a server with 10 cores, 256 GB RAM and an Asus GTX 1080 TI. Keras [31] was used for the NNs implementation and training. The code is not made publicly available as some of the content would disclose some of the industrial secret of our industrial partner.

3 Batch, retraining or incremental?

This section addresses the first methodological question of the paper, relative to the impact of the learning strategy (batch, retrained or incremental) on the detection accuracy. For this reason, we designed four algorithms to deal with the sequential setting:

1. `Batch` (Algorithm 1): the classifier is built once, never updated and applied to the testing set.
2. `Retrain`(F) (Algorithm 2): the classifier is retrained each F days, using two latest months and used to predict the incoming one. It is then discarded and a new one is built with the same procedure.
3. `Incremental`(F) (Algorithm 3): the classifier is updated (but not discarded) each F days, using two latest months and used to predict the incoming one.
4. `IncrementalEA`(F): this is an unsupervised approach based on an anomaly detection autoencoder [32].

Each batch B_i is made of the transactions of the i^{th} day. In the pseudo-codes, $batch_{cl}$ and $online_{cl}$ refer to the batch and online neural network (NN) implementation of the classifier.

To avoid biases related to the classifier structure, all NNs used in this paper share the same topology. This NN topology is the one currently used in production by our industrial partner Worldline. It is a dense neural network, selected after an extensive parameter tuning addressing the unbalancedness issue. In internal production setting, this NN was shown to outperform other state-of-the-art approaches for unbalanced data, like focal loss [33], class-balanced loss [34] and classical resampling strategies [35]. For the sake of confidentiality, no more details may be disclosed. For a wider discussion on unbalancedness in the specific case of fraud detection, please refer to [7].

The instances in Table 1 have been assessed in the first experimental session whose results are in Table 2.

The first conclusion of those experiments is that batch approaches (`Batch`, `Retrain(30)` and `Retrain(1)`) are outperformed by their incremental counterparts (Fig. 2). `Batch` and `Retrain(30)` are significantly less accurate than all incremental approaches. `Retrain(1)` is statisti-

cally equivalent to `Incremental(1)` but its training time is 180 times longer (see Table 2).

Our second conclusion is that the frequency of update of the incremental approaches (`Incremental(30)`, `Incremental(1)`) can be tuned to boost the performance, but the effect is limited. Overall, adopting an incremental approach increases the average AUPRC accuracy from around 6.7 to 10.3%.

As far as `IncrementalEA(1)` is concerned, the results are not encouraging. This is pretty much in line with results in our previous research [7,36] showing the inadequacy of pure unsupervised methods in fraud detection.

Figure 2 summarizes the previous results in the form of a *Friedman/Nemenyi* (F/N) test ([37]). All Friedman null hypotheses were rejected with $\alpha = 0.05$. For the Nemenyi post hoc test, a method is considered as significantly better than another if its mean rank is more than the critical difference CD higher (the higher, the better). Results for additional metrics are given in Appendix A.

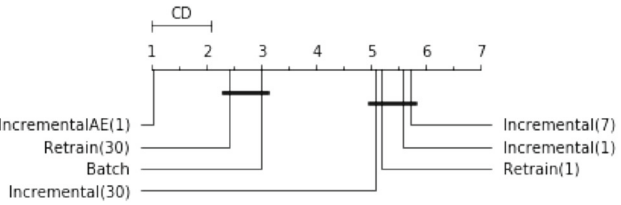

Fig. 2 Friedman–Nemenyi test based on the card-based AUPRC (70 days with one score per day). The plot compares batch approaches versus incremental approaches. Notice that the a priori fraud ratio based on cards is 0.201%. Each 70 results is averaged over five runs. The averages over the 5x70 individual results are reported on Table 2

Algorithm 1: Batch

initialize $batch_{cl}$ using classical batch (2 months of data);
$l_g \leftarrow$ length of the gap set (in days);
for *each new testing day* i **do**
  rank all transactions of B_i according to $batch_{cl}$;
  compute Pr@100 and AUPRC;
end

Algorithm 2: Retrain (F)

Input: F : the frequency of update (in days)
initialize $retrain_{cl}$ using classical batch (2 months of data);
$l_g \leftarrow$ length of the gap set (in days);
for *each new testing day* i **do**
  if F==*1* ***or*** F *days since last update* **then**
    % retrain the model, ignoring the gap set
    $B \leftarrow$ gather data B_{i-l_g-1} to B_{i-l_g-61} (2 months of data);
    delete $retrain_{cl}$;
    train $retrain_{cl}$ from scratch with B;
  end
  rank all transactions of B_i according to $online_{cl}$;
  compute Pr@100 and AUPRC;
end

Algorithm 3: Incremental (F)

Input: F : the frequency of update (in days)
initialize $online_{cl}$ using classical batch learning (2 months of data);
$l_g \leftarrow$ length of the gap set (in days);
for *each new testing day* i **do**
  if F==*1* ***or*** F *days since last update* **then**
    % update the model, ignoring the gap set
    $B \leftarrow$ gather newly available data from last update to now;
    update $online_{cl}$ with B as a new epoch;
  end
  rank all transactions of B_i according to $online_{cl}$;
  compute Pr@100 and AUPRC;
end

4 DTEL-like improvement of incremental learning

This section implements a DTEL-like enhancement of the incremental strategy used in the previous section. Some changes with respect to original DTEL are made:

1. Given the importance of ranking in the AUPRC and Pr@100 assessment, to compute diversity we replace the *Yule's Q-statistics* with *Spearman's rank correlation* and other correlation measures (e.g., *negative correlation*).
2. We change the transfer learning strategy according to our recent work on transfer learning for fraud detection [38] (Algorithm 4). Our transfer process is a nonlinear monotonous transformation[1] of the new data batch cumulative distribution (the source domain) into the cumulative distribution of the data used to initialize the incremental model (the target domain). It can be easily achieved by modifying the percentile normalization. (To make the paper self-contained, a short description is provided in Appendix B.)

[1] Though this applies to continuous features, discrete features may be re-encoded to get continuous features, see [39] for details.

Table 1 This table summarizes the considered approaches for the first part of this paper. More details about the strategies and hyperparameters can be found in Sect. 3

	Setting	Frequency	c classifier	Diversity	Transfer	Hyper param.
Batch	Batch	None	1	no	no	–
Retrain(30)	Retrain	30 days	1	no	no	–
Retrain(1)	Retrain	1 day	1	no	no	–
Incremental(30)	Update	30 days	1	no	no	–
Incremental(7)	Update	7 days	1	no	no	–
Incremental(1)	Update	1 day	1	no	no	–

Table 2 This table summarizes the results for the first part of this paper. Update time is an indicative execution time to update the model after each new data batch arrives. Each result is averaged over five runs. Notice that since the NN training is performed on GPU and data manipulation on CPU, the training part is not necessarily the most time-consuming part of the update. * Model is only updated each month or week. ** Model is never updated

	Mean AUPRC	Std AUPRC	Mean Pr@100	Std Pr@100	Update time
Batch	6.71 %	2.18 %	20.44 %	6.12 %	00.00 s **
Retrain(30)	6.53 %	2.25 %	20.18 %	6.21 %	91.48 s *
Retrain(1)	10.03 %	2.83 %	22.76 %	6.39 %	95.45 s
Incremental(30)	9.70 %	2.92 %	23.32 %	6.92 %	41.82 s *
Incremental(7)	10.29 %	2.81 %	23.00 %	6.18 %	09.64 s *
Incremental(1)	10.20 %	2.90 %	23.18 %	6.43 %	00.50 s
IncrementalAE(1)	2.62 %	1.01 %	9.24 %	3.8 %	00.38 s

3. Ensemble relies on the simple average of probability scores of 10 independently trained NN.

Algorithm 4: Incremental approach with transfer

$B_{init} \leftarrow$ 2 months of data for model initialization;
initialize $online_{cl}$ using classical batch learning and B_{init};
$g \leftarrow$ length of the gap set (in days);
for *each feature* f **do**
  $CDFtarget_f \leftarrow$ the CDF of feature f from B_{init};
end
for *each new testing day* i **do**
  % update the model, with transfer, ignoring the gap set
  gather the newly available data B_{i-g-1};
  % transfer B_{i-g-1} to match the distribution of $online_{cl}$;
  for *each feature* f **do**
    $CDFsource_f \leftarrow$ the CDF of feature f from B_{i-g-1};
    transfer by matching $CDFsource_f$ on $CDFtarget_f$;
  end
  if $F==1$ ***or*** F *days since last update* **then**
    update $online\_clf$ with B_{i-g-1} as a new epoch;
  end
  rank all transactions of B_i according to $online_{cl}$;
  compute Pr@100 ;
end

We implemented several variants of the incremental algorithm (Algorithm 3) (see Table 3)[2] by combining different ensemble, diversity and transfer learning aspects:

- E: Ensemble version of Algorithm 3.
- ED_R: variant of E where each NN of the ensemble is independently updated with probability p (hyperparameter calibrated by means of the validation set). After a few days, each NN has therefore trained on a different subset of days.
- ED_RT: variant of ED_R including the transfer learning step discussed above.
- ED_{Sp}: variant of E, but when the model is in the ranking phase, Spearman's correlations are computed among all the NN predictions and the more correlated NN is re-initialized at the end of the for loop. A variation consisting in re-initializing n_r NN(s) per day was considered. n_r must be tuned.
- ED_{Sp}T: variant of ED_{Sp} enhanced with transfer learning.
- ED_{cos}: variant of ED_{Sp}, using cosine similarity instead of Spearman's correlation
- ED_{ELM}: variant of E where the extreme learning [40] is considered as a source of diversity. Each of the NN is extended with n_l (to be tuned) layer(s) between the input and first dense, trainable, layer. The added layers are

[2] Acronyms in Table 3 follow this convention: "E" stands for ensemble of NNs, "D" for diversity criterion and "T" indicates we used transfer learning.

Table 3 This table summarizes the considered approaches of this paper. More details about the strategies and hyperparameters can be found in the main text

	Setting	Frequency	c classifier	Diversity	Transfer	Hyper param.
E	Update	1 day	10	No	No	–
ED_R	Update	1 day	10	Random	No	p
ED_RT	Update	1 day	10	Random	Yes	p
ED_{Sp}	Update	1 day	10	$\rho_{Spearman}$	No	n_r
ED_{Sp}T	Update	1 day	10	$\rho_{Spearman}$	Yes	n_r
ED_{cos}	Update	1 day	10	Cosine	No	n_r
ED_{ELM}	Update	1 day	10	Extreme LM	No	n_l
ED_{NC}	Update	1 day	10	Neg. Correl.	No	λ
ED_{NC}T	Update	1 day	10	Neg. Correl.	Yes	λ

Table 4 This table summarizes the results for the second part of this paper. Update time is an indicative execution time to update the model after each new data batch arrives. Each result is averaged over five runs. Notice that since the NN training is performed on GPU and data manipulation on CPU, the training part is not necessarily the most time-consuming part of the update

	Mean AUPRC	Std AUPRC	Mean Pr@100	Std Pr@100	Update time
E	11.07 %	3.21 %	24.40 %	7.17 %	14.78 s
ED_R	11.15 %	3.13 %	24.65 %	7.03 %	10.79 s
ED_RT	11.13 %	3.20 %	24.69 %	7.12 %	13.46 s
ED_{Sp}T	7.70 %	**2.30** %	19.46 %	5.51 %	47.07 s
ED_{Sp}T	7.63 %	2.31 %	19.26 %	**5.45** %	42.86 s
ED_{cos}	8.31 %	2.58 %	20.06 %	5.76 %	48.00 s
ED_{ELM}	9.41 %	2.94 %	21.28 %	6.30 %	65.65 s
ED_{NC}	11.61 %	3.52 %	**26.22** %	8.11 %	21.34 s
ED_{NC}T	**11.63** %	3.51 %	26.20 %	8.24 %	20.09 s

randomly initialized, untrainable and share the same size as the largest layer of Worldline's industrial NN. Linear activation function was selected, by trial and error.

– ED_{NC}: variant of E with a *negative correlation* (NC) loss function containing a term promoting the diversity in the ensemble.

Negative correlation (NC) learning [41] introduces a penalty term into the loss function of each NN in the ensemble so that all the NNs can be trained simultaneously and interactively on the same data. The error function $\mathcal{E}_i$ for NN i in NC learning is defined by:

$$\mathcal{E}_i = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}(F_i(n) - t(n))^2 + \frac{\lambda}{N}\sum_{n=1}^{N} p_i(n) \qquad (1)$$

where $F_i(n)$ is the output of NN i for sample n, $t(n)$ is the target value for sample n and N is the number of NN in the ensemble. The first term is the empirical risk function of NN i. The second term with $p_i(n) = (F_i(n) - F(n))\sum_{i\neq j}(F_j(n) - F(n))$ is a correlation penalty function, with $F(n)$ the output of ensemble of NNs.

– ED_{NC}T: variant of ED_{NC} with a transfer learning step.

Note that transfer learning can be added to all possible approaches. To avoid to report and analyze every approach twice, we choose to include only the transfer learning variants for a subset of the approaches of this study.

Table 4 is used to summarize the 70 evaluations (one per test day) of the metric for each approach. We report the mean and standard deviation for the AUPRC and the Pr@100. To avoid variability in the NN training, each result is the mean of five different initializations. Most of the approaches are already stabilized as they imply ensembles of classifiers. Notice that the variability is still high in practice: This is due to the number of frauds to detect which can be very different each day and is not due to the initialization of the NNs. A calibrated version of the AUPRC, which removes the effect of the varying proportion of fraud to detect, is presented in Appendix A.

Table 4 also reports the mean execution time for updating the models. Notice that most models are updated daily, but some are updated less frequently. Also, since the NN training is performed on GPU and data manipulation on CPU, the training part is not necessarily the most time-consuming part of the update.

Figure 3 summarizes the results in the form of a *Friedman/Nemenyi* (F/N) test ([37]). From this figure, it appears that:

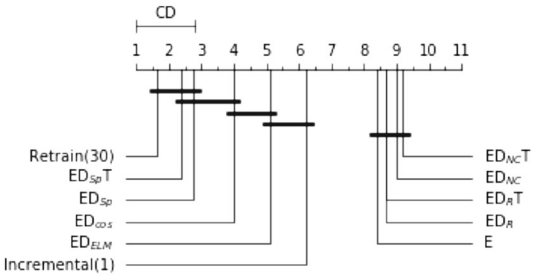

Fig. 3 Friedman–Nemenyi test based on the card-based AUPRC (70 days with one score per day). The plot compares various ensembles and baseline approaches. Notice that the a priori fraud ratio based on cards is 0.201%. Each of the 70 results is averaged over five runs. The averages over the 5x70 individual results are reported on Table 4

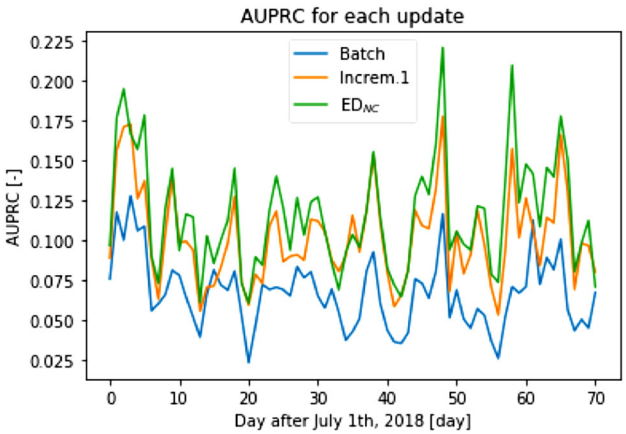

Fig. 4 The two sets of experiments show an increase in performance from around 6.7% average AUPRC to 11.6% average AUPRC

– The ensemble approach `E` significantly outperforms `Incremental(1)`, yet at the price of a longer training time.
– Adding diversity is sometimes beneficial (ED$_R$,ED$_{NC}$) and sometimes not (ED, ED$_{ELM}$, ED$_{cos}$). In the beneficial case, the gain is limited but the training time is not really impacted.
– Considering transfer learning does not make a big difference. Consider ED$_{Sp}$ vs. ED$_{Sp}$T, ED$_R$ vs. ED$_R$T and ED$_{NC}$ vs. ED$_{NC}$T. The mean ranks are really close and the F/N test concludes it is a draw in the three cases (this was the case even for unreported pairs). Our conclusion is to use either ED$_R$ or ED$_{NC}$ with no transfer, except if the transfer is justified by a lack of source data (as in [39]).

This second set of experiments allows us to increase the performance from around 10.3% average AUPRC (at the end of the first set of experiments) to 11.6% average AUPRC (see Fig. 4 to visualize the results as a function of time).

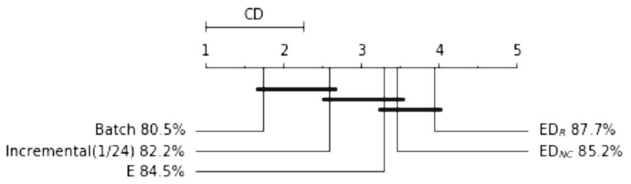

Fig. 5 Friedman–Nemenyi test based on the ROC AUC (24 hours with one score per hour). The plot compares the main approaches on the public dataset. Each of the 24 results is averaged over ten runs; average over the 10x24 individual results is reported besides the method name

5 Assessment on public data

Given the confidential nature of the dataset, it is interesting to assess the quality of our approach on a public dataset. In this section we make use of the Kaggle "Credit Card Fraud Detection" dataset [29], made available some years ago by our research group to support the reproducibility efforts in the fraud detection community [32]. Note that this is one of the few fraud detection datasets publicly available and one of the most accessed datasets on the Kaggle platform (more than seven million views so far). This dataset is a two-day long, anonymized extract from the same process underlying the private dataset. Because of the limited time window, we have to adapt the time frames: The first day is used for training and each chunk of data refers to one hour of transactions. Also, since no card holder information is available, we may only report accuracy in terms of area under the ROC curve.

Figure 5 reports the outcome of the *Friedman/Nemenyi* test. For simplicity, and to ensure sufficient statistical power to the test, we limit ourselves to report the most important methods and baselines from previous sections. In spite of the smaller time frame, the trend of Figs. 2 and 4 is confirmed.

6 Conclusion

Incremental learning is one of the oldest learning strategies in AI. However, few evidence exists about the current utilization of this strategy in fraud detection production environments. Nowadays, the interest in adopting incremental approaches is raising also for nonanalytical reasons, e.g., the wish to avoid storing sensitive data and being exposed to infringements of the European GDP Regulation.

This paper shows that incremental learning is a competitive alternative to conventional batch learning settings for fraud detection in real-world transactions streams. Though the case study is limited to five months of data on e-commerce transactions, we believe that other fintech domains can benefit from our findings.

The paper discusses, implements and assesses 16 approaches (on private and public data) and answers to two questions: (i) Is incremental learning viable for fraud detection? and

(ii) Are there variants of incremental learning (notably based on ensemble learning, diversity and transfer learning) which may still improve its accuracy?

The answer to the first question is affirmative. The second is more mitigated, though the adoption of proper diversity measures in ensembles (notably negative correlation) appears to improve the accuracy.

Future work will focus on extending the set of considered approaches (alternative transfer learning and diversity measures). Also we intend to study the risk of catastrophic forgetting in the adoption of techniques to counter concept drift.

Funding This work was supported by the TeamUp DefeatFraud project funded by Innoviris (2017-R-49a), Brussels, Belgium. We thank this agency for allowing us to conduct both fundamental and applied research. B. Lebichot also thanks LouRIM, Université catholique de Louvain, Belgium, for their support.

Declarations

Conflicts of interest The authors declare they have no conflicts of interest or competing interests.

Availability of data and material The main dataset cannot be made available for confidential reasons. A good proxy is the public Kaggle dataset [29], a two-day long, anonymized extract from the same process. Experimental results with the public dataset are reported in Sect. 5.

Code availability Code cannot be made available for confidential reasons.

A Appendix: Additional metrics

As explained in Sect. 2, taking into account the Pr@100 can be more relevant because of the limited number of investigators or investigations per day. Figures 6 and 7 summarize the results in the form of F/N tests ([37]).

Another interesting indicator can be obtained by calibrating the AUPRC. Indeed, the number of fraud to detect per day ranges from 0.095 to 0.335% (the coefficient of variation is 21%.). The AUPRC is calibrated in such a way that it is invariant to the fraud prior (see [42] for details). Figures 8 and 9 summarize the results in the form of F/N tests ([37]).

The purpose of those two additional metrics is to be extensive on the results and to provide different points of view for the evaluation. For the sake of conciseness, we will not comment all the results once again. This appendix shows that the conclusions are the same, in terms of statistical tests, regardless of the metric used: Pr@100, uncalibrated AUPRC and calibrated AUPRC. However, since those three metrics do not measure exactly the same quantities, there are small variations in the actual ordering of the methods in terms of average results.

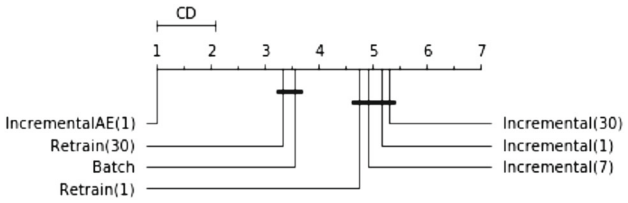

Fig. 6 F/N test based on the metric Pr@100. See caption of Fig. 2 for details

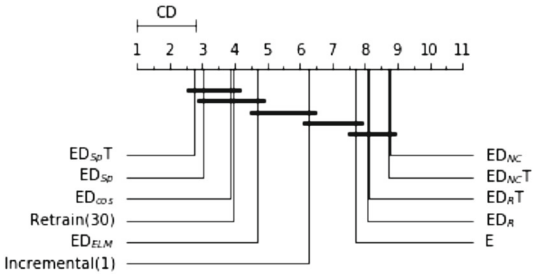

Fig. 7 F/N test based on the metric Pr@100. See caption of Fig. 3 for details

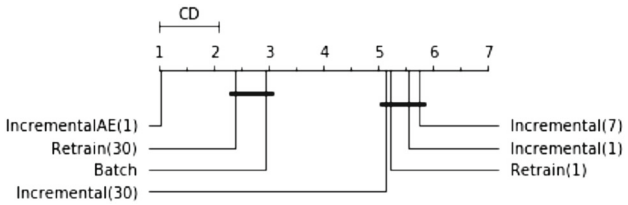

Fig. 8 F/N test based on the metric calibrated AUPRC. See caption of Fig. 2 for details

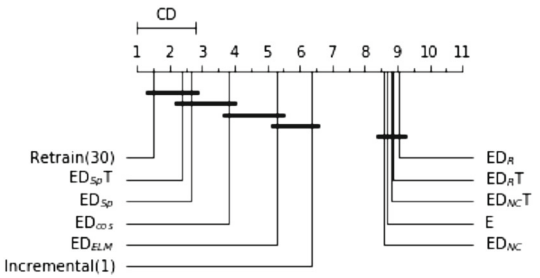

Fig. 9 F/N test based on the metric calibrated AUPRC. See caption of Fig. 3 for details

B Appendix: Transfer learning in details

This description is based on [38] and is here to make the paper self-contained. Algorithm 4 details how transfer learning is embedded into the continuous approach. In our case, the target domain is the data used to initialize the incremental NN model and the source domain is the new batch of data. We consider here only the univariate case where each feature is transferred independently of the others.

The transfer process is a nonlinear monotonous transformation of the values of a continuous random variable X (the

source data), such that the cumulative distribution function (CDF) of X after transformation matches a given CDF F (the target data).

First, we compute the value of the empirical CDF of X (noted $\hat{F}$) at each observed value $x_i, i = 1, \ldots, n$. The transferred value x_i' is then chosen such that $F(x_i') = \hat{F}(x_i)$. We denote source examples by $x_i^{(s)}, i = 1, \ldots, n^{(s)}$ and target examples by $x_j^{(t)}, j = 1, \ldots, n^{(t)}$, with $n^{(s)}$ and $n^{(t)}$, respectively, the number of source and target examples. We also note the value of the empirical CDF as $p_i^{(s)} = \hat{F}^{(s)}(x_i^{(s)})$ and $p_j^{(t)} = \hat{F}^{(t)}(x_j^{(t)})$.

The source examples $x_i^{(s)}$ are transformed to a CDF that matches the empirical CDF $\hat{F}^{(t)}$ of the target examples. The target examples are left unmodified. For each source example $x_i^{(s)}$ and the corresponding empirical CDF value $p_i^{(s)}$, we find the two consecutive empirical CDF values $p_{j_1}^{(t)}$ and $p_{j_2}^{(t)}$ framing $p_i^{(s)}$ in the target domain:

$$p_{j_1}^{(t)} \leq p_i^{(s)} < p_{j_2}^{(t)}$$

with $j_1 + 1 = j_2$. $x_i^{(s)\prime}$ is then computed as the linear interpolation between the values $x_{j_1}^{(t)}$ and $x_{j_2}^{(t)}$.

References

1. HSN Consultants, Inc, The nilson report 2019 (consulted on 2020-03-17). https://nilsonreport.com
2. Abdallah, A., Maarof, M.A., Zainal, A.: Fraud detection system?: A survey. J. Netw. Comput. Appl. **68**, 90–113 (2016)
3. Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, J.C.: Data mining for credit card fraud: A comparative study. Decis. Support Syst. **50**(3), 602–613 (2011)
4. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. learn. **23**(1), 69–101 (1996)
5. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent, In: Proceedings of the 20th international conference on machine learning (icml-03), pp. 928–936 (2003)
6. Hazan, E., Agarwal, A., Kale, S.: Logarithmic regret algorithms for online convex optimization. Mach. Learn. **69**(2–3), 169–192 (2007)
7. Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., Bontempi, G.: Learned lessons in credit card fraud detection from a practitioner perspective. Expert Syst. Appl. **10**(41), 4915–4928 (2014)
8. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. Inf. Fusion **6**(1), 5–20 (2005)
9. Sun, Y., Tang, K., Zhu, Z., Yao, X.: Concept drift adaptation by exploiting historical knowledge. IEEE Trans. Neural Netw. Learn. Syst **29**(10), 4822–4832 (2018)
10. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010)
11. W. N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001, pp. 377–382
12. Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Trans. Neural Netw. **22**(10), 1517–1531 (2011)
13. S. Ghosh, D. L. Reilly, Credit card fraud detection with a neural-network, in: System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on, Vol. 3, IEEE, 1994, pp. 621–630
14. Dorronsoro, J.R., Ginel, F., Sgnchez, C., Cruz, C.S.: Neural fraud detection in credit card operations. IEEE Trans. Neural Netw. **8**(4), 827–834 (1997)
15. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Netw. (2019)
16. Fu, K., Cheng, D., Tu, Y., Zhang, L.: Credit card fraud detection using convolutional neural networks, In: International Conference on Neural Information Processing, Springer, pp. 483–490 (2016)
17. Pumsirirat, A., Yan, L.: Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. Int. J. Adv. Comput. Sci. Appl. **9**(1), 18–25 (2018)
18. Abakarim, Y., Lahby, M., Attioui, A.: An efficient real time model for credit card fraud detection based on deep learning, In: Proceedings of the 12th International Conference on Intelligent Systems: Theories and Applications, pp. 1–7 (2018)
19. Nguyen, T.T., Tahir, H., Abdelrazek, M., Babar, A.: Deep learning methods for credit card fraud detection, arXiv preprint arXiv:2012.03754 (2020)
20. Najadat, H., Altiti, O., Aqouleh, A.A., Younes, M.: Credit card fraud detection based on machine and deep learning, In: 2020 11th International Conference on Information and Communication Systems (ICICS), IEEE, pp. 204–208 (2020)
21. Forough, J., Momtazi, S.: Ensemble of deep sequential models for credit card fraud detection. Appl. Soft Comput. **99**, (2021)
22. Alippi, C., Boracchi, G., Roveri, M.: Just-in-time classifiers for recurrent concepts. IEEE Trans. Neural Netw. Learn. Syst. **24**(4), 620–634 (2013)
23. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. (CSUR) **46**(4), 1–37 (2014)
24. Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., Bontempi, G.: Scarff: a scalable framework for streaming credit card fraud detection with spark. Inf. Fusion **41**, 182–194 (2018)
25. Saito, T., Rehmsmeier, M.: The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PloS one **10**(3), (2015)
26. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves, In: Proceedings of the 23rd international conference on Machine learning, pp. 233–240 (2006)
27. Lebichot, B., Braun, F., Caelen, O., Saerens, M.: A graph-based, semi-supervised, credit card fraud detection system, pp. 721–733. Springer, Cham (2017)
28. Dal Pozzolo, A.: Adaptive machine learning for credit card fraud detection, Ph.D. thesis, Universite Libre de Bruxelles (2015)
29. Machine Learning Group - ULB, Credit card fraud detection (consulted on 2020-06-28). https://www.kaggle.com/mlg-ulb/creditcardfraud
30. Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P.-E., He, L., Caelen, O.: Sequence classification for credit-card fraud detection. Expert Syst. Appl. **100**, 234–245 (2018)
31. Chollet, F.: et al., Keras, https://keras.io (2015)
32. R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, arXiv preprint arXiv:1901.03407 (2019)
33. Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection, In: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988 (2017)
34. Cui, Y., Jia, M., Lin, T.-Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples, In: Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition, pp. 9268–9277 (2019)
35. Japkowicz, N.: Learning from imbalanced data sets: a comparison of various strategies, In AAAI Workshop on Learning from Imbalanced Data Sets (2000)
36. Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., Bontempi, G.: Combining unsupervised and supervised learning in credit card fraud detection. Inf. Sci. **557**, 317–331 (2019)
37. Demsar, J.: Statistical comparaison of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
38. B. Lebichot, T. Verhelst, Y.-A. Le Borgne, L. He-Guelton, F. Oblé, G. Bontempi, Transfer learning strategies for credit card fraud detection (submitted for publication)
39. Lebichot, B., Le Borgne, Y.-A., He-Guelton, L., Oblé, F., Bontempi, G.: Deep-learning domain adaptation techniques for credit cards fraud detection. In: Oneto, L., Navarin, N., Sperduti, A., Anguita, D. (eds.) Recent Advances in Big Data and Deep Learning, pp. 78–88. Springer International Publishing, Cham (2020)
40. Huang, J., Smola, A.J., Gretton, A., Borgwardt, K.M., Scholkopf, B.: Correcting sample selection bias by unlabeled data, In: Proceedings of the 19th International Conference on Neural Information Processing Systems, NIPS'06, MIT Press, pp. 601–608 (2006)
41. Liu, Y., Yao, X.: Ensemble learning via negative correlation. Neural Netw. **12**(10), 1399–1404 (1999)
42. Siblini, W., Fréry, J., He-Guelton, L., Oblé, F., Wang, Y.-Q.: Master your metrics with calibration, In: International Symposium on Intelligent Data Analysis, Springer, pp. 457–469 (2020)