Information Systems Frontiers (2020) 22:1113–1131
https://doi.org/10.1007/s10796-020-10022-7

The Effects of Data Sampling with Deep Learning and Highly Imbalanced Big Data

Justin M. Johnson[1] · **Taghi M. Khoshgoftaar**[1]

Published online: 21 June 2020

Abstract

Training predictive models with class-imbalanced data has proven to be a difficult task. This problem is well studied, but the era of *big data* is producing more extreme levels of imbalance that are increasingly difficult to model. We use three data sets of varying complexity to evaluate data sampling strategies for treating high class imbalance with deep neural networks and big data. Sampling rates are varied to create training distributions with positive class sizes from 0.025%–90%. The area under the receiver operating characteristics curve is used to compare performance, and thresholding is used to maximize class performance. Random over-sampling (ROS) consistently outperforms under-sampling (RUS) and baseline methods. The majority class proves susceptible to misrepresentation when using RUS, and results suggest that each data set is uniquely sensitive to imbalance and sample size. The hybrid ROS-RUS maximizes performance and efficiency, and is our preferred method for treating high imbalance within big data problems.

Keywords Class imbalance · Big data · Data sampling · Artificial neural networks · Deep learning

1 Introduction

Class imbalance occurs when one class, the majority group, is significantly larger than the opposing minority class. This phenomenon arises in many critical industries, e.g. medical (Rao et al. 2006), financial (Wei et al. 2013), and environmental (Kubat et al. 1998). In these examples, the minority group is also the positive class, or the class of interest. When class imbalance exists within training data, learners will typically over-classify the majority group due to its increased prior probability. As a result, the instances belonging to the minority group are misclassified more often than those belonging to the majority group. Furthermore, some evaluation metrics, such as accuracy, may mislead an analyst with high scores that incorrectly indicate good performance. For example, given a binary data set with a positive class size of just 1%, a simple learner that always outputs the negative class will score 99% accuracy. Learning from these class-imbalanced data sets can be very difficult, and advanced learning methods are usually required to obtain meaningful results.

In this paper, we denote the level of class imbalance in a binary data set using the ratio of negative samples to positive samples, i.e. $n_{neg} : n_{pos}$. For example, the imbalance of a data set with 400 negative samples and 100 positive instances is denoted by 80:20. Equivalently, we sometimes refer to a binary data set using just the size of the positive class, e.g. 20% positive class size. In Section 2, we adopt the notation of Buda et al. (Buda et al. 2018) and denote levels of imbalance using ρ (Eq. 1), where $\max_i\{|C_i|\}$ is the size of the largest class and $\min_i\{|C_i|\}$ is the size of the smallest class. In other words, some authors in related works choose to denote an 80:20 imbalance using $\rho = 4$.

$$\rho = \frac{\max_i\{|C_i|\}}{\min_i\{|C_i|\}} \tag{1}$$

Generally, model performance degrades as the level of class imbalance increases and the relative size of the positive class decreases (Buda et al. 2018). We classify a data set as highly imbalanced when the size of the positive class makes up less than 1% of a binary data set. When the total number of positive occurrences becomes even more infrequent, we describe the data set as exhibiting rarity. Weiss (2004) distinguish between absolute rarity and

✉ Justin M. Johnson
jjohn273@fau.edu

Taghi M. Khoshgoftaar
khoshgof@fau.edu

[1] Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA

relative rarity. Absolute rarity occurs when there are very few samples for a given class, regardless of the size of the majority class. Unlike absolute rarity, a relatively rare class can make up a small percentage of a data set and still have many occurrences when the overall data set is very large. For example, given a data set with 10 million records and a relative positive class size of 1%, there are still 100,000 positive cases to use for training machine learning models. Therefore, we can sometimes achieve good performance with relatively rare classes, especially when working with large volumes of data. The unsampled data sets used in this study have positive class sizes between 0.03%–0.20% and can be safely regarded as highly imbalanced data sets with relatively rare positive classes.

The challenges of working with class-imbalanced data tend to be compounded when working in the context of big data (Bauder et al. 2018). In general, the concept of big data refers to data which exceeds the capabilities of standard data storage and processing. These data sets are also defined using the four Vs: volume, variety, velocity, and veracity (Dumbill 2012; Ahmed 2014). The large volumes of data being collected require highly scalable hardware and efficient analysis tools, often demanding distributed implementations. In addition to adding architecture and network overhead, distributed systems have been shown to exacerbate the negative effects of class imbalanced data (Leevy et al. 2018). The variety of big data corresponds to the mostly unstructured, diverse, and inconsistent representations that arise as data is consumed from multiple sources over extended periods of time. Advanced techniques for quickly processing incoming data streams and maintaining appropriate turnaround times are required to keep up with the rate at which data is being generated, i.e. data velocity. Finally, the veracity of big data, i.e. its accuracy and trustworthiness, must be regularly validated to ensure results do not become corrupted. We use three big data classification problems to evaluate data-level techniques for addressing class imbalance.

The first two data sets, Medicare Part B and Part D, summarize claims that medical providers have submitted to Medicare and include a class label that indicates whether or not the provider is known to be fraudulent. Medicare is a United States healthcare program that provides affordable health insurance to individuals 65 years and older, and other select individuals with permanent disabilities (https://www.medicare.gov/). An increasing population coupled with the digitization of patient records has produced an abundance of data (Kankanhalli et al. 2016). The Part B and Part D data sets each have 4.6 and 3.6 million observations, respectively, with positive class sizes $< 0.04\%$. They were made publicly available by the Centers for Medicare and Medicaid Services (CMS) in order to increase transparency and reduce fraud. The third data set comes from the network security domain and contains more than 3 million samples of network traffic from a production environment server. Samples within this Denial of Service (DoS) data set are labeled as either "normal" or "attack", and the positive attack class makes up just 0.20% of the entire data set. A DoS attack occurs when a malicious user attempts to consume all of a target server's resources in order to block valid users from accessing the target's services. Each of these binary classification problems qualifies as big data problems with high class imbalance. They are described in full in Section 3.

Data-level and algorithm-level techniques for addressing class imbalance have been studied extensively over the last two decades. Data methods use sampling to alter the distribution of the training data, effectively reducing the level of class imbalance for model training. Random over-sampling (ROS) and random under-sampling (RUS) are the two simplest data-level methods for addressing class imbalance. ROS increases the size of the minority class by randomly copying minority samples, and RUS decreases the size of the majority class by randomly discarding samples from the majority class. From these fundamental data-level methods, many more advanced variants have been developed (Wilson 1972; Chawla et al. 2002; Han et al. 2005; Jo and Japkowicz 2004). Related works suggest that these advanced data-level methods do not always improve performance (Leevy et al. 2018; Hasanin et al. 2019). Therefore, we use plain random sampling strategies to determine how levels of imbalance and sample sizes affect performance in this study, and we leave the evaluation of more intelligent data-level techniques for future work. Algorithm-level methods modify the training process to increase the impact of the minority class and reduce bias towards the majority class. Direct algorithm-level methods modify a machine learning algorithm's underlying learner, usually by incorporating class costs or weights, while meta-learner methods use a wrapper to convert non-cost-sensitive learners into cost-sensitive learners (Ling and Sheng 2007). Finally, there are a number of hybrid methods that combine two or more data-level methods and algorithm-level methods (He and Garcia 2009; Liu et al. 2009; Chawla et al. 2003; Sun 2007; Khoshgoftaar et al. 2014). In this study, we focus specifically on the effect that data sampling has on training neural networks with highly imbalanced big data.

While these data-level methods have been combined with traditional learners to treat class imbalance in both big and non-big data problems, we found there is little work that properly evaluates them using artificial neural network (ANN) models. Deep learning is a sub-field of machine learning that uses the ANN with two or more hidden layers to approximate some function f^*, where f^* can be used to map input data to new representations

or make predictions (Goodfellow et al. 2016). The ANN, inspired by the biological neural network, is a set of interconnected neurons, or nodes, where connections are weighted and each neuron transforms its input into a single output by applying a non-linear activation function to the sum of its weighted inputs. In a feedforward network, input data propagates through the network in a forward pass, each hidden layer receiving its input from the previous layer's output, producing a final output that is dependent on the input data, the choice of activation function, and the weight parameters (Witten et al. 2016). Gradient descent optimization adjusts the network's weight parameters in order to minimize the loss function, i.e. the error between expected output and actual output. Composing multiple non-linear transformations creates hierarchical representations of the input data, increasing the level of abstraction through each transformation. The deep learning architecture, i.e. deep neural network (DNN), achieves its power through this composition of increasingly complex abstract representations (Goodfellow et al. 2016). Deep learning methods have proven very successful in solving complex problems related to natural language and vision (LeCun et al. 2015). These recent successes can be attributed to an increased availability of data, improvements in hardware and software (Abadi et al. 2015; Theano Development Team 2016; Chollet and et al. 2015; Paszke et al. 2017; Chetlur et al. 2014), and various algorithmic breakthroughs that speed up training and improve generalization to new data (Krizhevsky et al. 2012). We are interested in using these deep models to solve complex classification tasks that exhibit high class imbalance and rare classes. We use both two- and four-hidden-layer networks when evaluating sampling methods in this study.

The primary contribution of this study is in providing a comprehensive evaluation of multiple sampling strategies for training DNN models with highly imbalanced big data. We compare ROS, RUS, and a hybrid ROS-RUS method across three data sets with varying levels of difficulty. Sampling rates are varied to create 17 training distributions with class imbalance levels ranging from 0.025%–90.0%. For each data set, distribution, and architecture set, we train 30 models and report the average area under the Receiver Operating Characteristic curve (ROC AUC) (Provost and Fawcett 1999) score on a 20% holdout test set. Tukey's HSD (honestly significant difference) (Tukey 1949) test is used to estimate the significance of AUC results and identify the best performing methods. AUC results show that ROS and ROS-RUS significantly outperform RUS in nearly all cases. The RUS methods outperform baseline models under certain conditions, but experiences degradation when class imbalance levels are too high or negative class sample sizes are too small. Additionally, RUS results reveal that each data set is uniquely sensitive to class imbalance and sample size, e.g. Part B is very sensitive while DoS performance is relatively stable. After taking training times into consideration, we determine that ROS-RUS methods are the preferred method for treating high class imbalance within big data. Finally, we compare thresholded results to non-thresholded results and show that the default classification threshold of 0.5 is rarely optimal when training data is imbalanced. To the best of our knowledge, this is the first study to evaluate ROS, RUS, and ROS-RUS across multiple big data problems containing relatively rare positive classes.

The remainder of this paper is outlined as follows. Section 2 discusses several methods for training deep models with class-imbalanced data and other related works that have used the Medicare and DoS data sets. In Section 3 and Section 4 we describe the data sets used in this study and present our experiment design. Results are discussed in Section 5, and Section 6 closes with suggestions for future works.

2 Related Work

This section summarizes works that include data sampling methods for training DNN classifiers with imbalanced data. In addition, we mention other works that have used the Medicare and DoS data sets.

2.1 Deep Learning and Data Sampling

Anand et al. (1993) studied how class imbalance affected backpropagation and optimization using shallow neural networks. During optimization, the network loss and gradients are dominated by the majority class and the minority group has very little influence on weight updates. This tends to reduce the error of the majority group very quickly during early iterations while consequently increasing the error of the minority group. The related works that we present in this section balance the optimization process by modifying the distribution of the training data. Readers interested in algorithm-level methods for addressing class imbalance with neural networks should refer to our recent survey (Johnson and Khoshgoftaar 2019b).

Masko and Hensman (2015) explored the effects of ROS on class-imbalanced image data generated from the CIFAR-10 (http://www.cs.toronto.edu/~kriz/cifar.html) data set. The authors generated ten imbalanced distributions by varying the class sizes and create a maximum imbalance ratio of $\rho = 2.3$. Any class that is smaller than the largest class is treated as a minority and the minority classes are randomly over-sampled until balance is achieved. Results show that

the models trained with ROS perform almost as well as the baseline models that were trained with the original balanced distributions. While the results demonstrate that ROS is effective, we are more interested in problems containing very high levels of class imbalance.

Lee et al. (2016) combined RUS with transfer learning to classify highly imbalanced data sets of plankton images, WHOI-Plankton (Orenstein et al. 2015). This data set exhibits big data and rare class characteristics, with 3.4 million images spread over 103 classes and several classes making up less than 0.1% of the data set. The proposed two-phase learning procedure pre-trains a CNN with thresholded data and then fine-tunes the model using all available data. The thresholded data set for pre-training is constructed by randomly under-sampling the large classes down to 5,000 observations. This allows the minority group to contribute more to the gradient during pre-training and then takes advantage of all data during the fine-tuning phase. Two-phase learning was compared to six alternative methods and results show that it is effective in increasing the performance of the minority class without sacrificing that of the majority. While Lee et al. have found two-phase learning to be effective, another related work has shown that ROS outperforms two-phase learning. Therefore, we do not include two-phase learning in this study.

Buda et al. (2018) compare plain ROS and RUS with the two-phase learning method, and they use both ROS and RUS to balance the pre-training data in order to determine which is more effective. The authors evaluate these methods by simulating class imbalance ratios of $\rho \in [10, 100]$ on the MNIST (LeCun and Cortes 2010), CIFAR-10, and ImageNet (Russakovsky et al. 2015) data sets. The AUC is used to compare performance across a range of imbalance levels, and thresholding is used to maximize accuracy. Results show that ROS outperforms the baseline and RUS methods in nearly all case, thresholding improves accuracy, and RUS generally performs poorly. We follow a similar procedure and use the AUC to compare sampling methods and apply thresholding to maximize class performance. Unlike Buda et al., we use structured data sets instead of images and we use significance testing to identify the best methods. Furthermore, we recognize the training costs associated with ROS, and include a hybrid ROS-RUS method that improves efficiency.

The loading and processing of big data sometimes requires a distributed environment, e.g. MapReduce (Dean and Ghemawat 2008), Apache Spark (Zaharia et al. 2010), or Apache Storm (Requeno et al. 2019). Leevy et al. (2018) summarize the challenges and solutions that are associated with training models in these distributed environments. Fernández et al. (2017) provide additional details on training machine learning algorithms with class-imbalanced data using the MapReduce framework. Neural networks can be trained in a distributed manner using either data parallelism or model parallelism techniques (Chahal et al. 2019; Kennedy et al. 2019). We do not consider these environments in our study, however, because data sampling can be executed globally during pre-processing or at the batch level during training. Therefore, the sampling strategies used throughout this study will generalize to a distributed platform with similar results.

This study expands on related works by focusing specifically on high class imbalance and big data. Neural networks are trained on three large data sets, each with millions of records and imbalance levels of $\rho > 3000$. We use the AUC, Geometric Mean (G-Mean), training times, and statistical analysis to compare more than 20 sampling distributions using two network architectures.

2.2 Medicare Fraud Detection

Our research group has performed extensive research in the area of detecting anomalous provider behavior with CMS Medicare data. Bauder and Khoshgoftaar (2016b) proposed an outlier detection method that uses Bayesian inference to identify outliers, and successfully validated their model using claims data of a known Florida provider that was under criminal investigation for excessive billing. This experiment used a subset of 2012–2014 Medicare Part B data that included dermatology and optometry claims from Florida office clinics. In another study, Bauder and Khoshgoftaar (Bauder and Khoshgoftaar 2016a) use a subset of the 2012–2013 Medicare Part B data, i.e. Florida claims only, to model expected amounts paid to providers for services rendered to patients. The authors flag potential fraudulent providers by comparing actual payment amount deviations to the expected payment amounts using five different regression models. Another paper by Bauder et al. (2016) uses a Naive Bayes classifier to predict provider specialty types, and then flag providers that are practicing outside their expected specialty type as fraudulent. This study also used a Florida-only subset of 2013 Medicare Part B claims data, but unlike the previous work this experiment included all 82 provider types. The authors conclude that specialties with unique billing procedures, e.g. audiologist or chiropractic, are able to be classified with high precision and recall. Herland et al. (2017) expanded upon the work from Bauder et al. (2016) by incorporating 2014 Medicare Part B data and real-world fraud labels defined by the List of Excluded Individuals and Entities (LEIE) (Office of Inspector General 2019) data set. A Naive Bayes classifier is used to predict a provider's specialty type, and providers that are misclassified are labeled as potentially fraudulent. The authors find that grouping similar specialty types, e.g. Ophthalmology and Optometry, improves overall performance. Bauder and Khoshgoftaar (2018) merge the

2012–2015 Medicare Part B data sets, label fraudulent providers using the LEIE data set, and compare multiple traditional machine learning classifiers. Class imbalance is addressed with RUS, and various class distributions are generated to identify the optimal imbalance ratio for training. The C4.5 decision tree and logistic regression (LR) learners significantly outperform the support vector machine (SVM), and the 80:20 class distribution is shown to outperform 50:50, 65:35, and 75:25 distributions. These studies jointly show that the Medicare Part B claims data contains sufficient variability to detect bad actors and that the LEIE data set can be reliably used for ground truth fraud labels.

The Medicare experiments in this study are most closely related to works by Herland et al. (2018) and Herland et al. (2019). In one study, Herland et al. (2018) used Medicare Part B, Part D, and DMEPOS claims data from the years 2012–2016. Cross-validation and ROC AUC scores are used to compare LR, Random Forest (RF), and Gradient Boosted Tree (GBT) learners. Results show that the Part B data sets score significantly better on ROC AUC than the Part D data set, and the LR learner outperforms the GBT and RF learners with a max AUC of 0.816. In a second paper, Herland et al. (2019) use these same Medicare data sets to study the effect of class rarity with LR, RF, and GBT learners. In this study, the authors create an absolutely rare positive class by using subsets of the positive class to create new training sets. They reduce the positive class size to 1000, 400, 200, and 100, and then use RUS methods to treat imbalance and compare AUC scores. Results show that smaller positive class counts degrade model performance, and the LR learner with a RUS distribution of 90:10 performs best.

In a previous study, we evaluated the use of data sampling and deep learning using the Medicare Part B data set (Johnson and Khoshgoftaar 2019a). This previous work used data sampling methods to create training sets with positive class sizes in the range 1%–60%. We expand on prior work and focus specifically on the effects of data sampling with high class imbalance and big data by incorporating two new data sets with millions of records and relatively rare positive classes. Furthermore, we increase the range of positive class sizes to 0.025%–90% to better illustrate how class imbalance, sample size, and problem complexity affect performance. To the best of our knowledge, we are also the first to include recently released 2017 Medicare data in our experiments.

Several other research groups have used the CMS Medicare and LEIE data to identifying patterns, anomalies, and potentially fraudulent providers. Feldman and Chawla (2015) looked for anomalies in the relationship between medical school training and the procedures that physicians perform in practice by linking 2012 Medicare Part B data with provider medical school data obtained through the CMS physician compare data set (Centers for Medicare & Medicaid Services 2019b). Significant procedures for each school were used to evaluate school similarities and present a geographical analysis of procedure charges and payment distributions. Ko et al. (2015) used the 2012 CMS data to analyze the variability of service utilization and payments. The authors found that the number of patient visits is strongly correlated with Medicare reimbursement, and concluded that there is a possible 9% savings within the field of Urology alone. Chandola et al. (2013) use claims data and fraud labels from the Texas Office of Inspector General's exclusion database to detect anomalies. They confirm the importance of including provider specialty types in fraud detection, showing that the inclusion of specialty attributes increases AUC scores from 0.716 to 0.814. Branting et al. (2016) propose a graph-based method for estimating healthcare fraud risk within the 2012–2014 CMS and LEIE data sets. The authors leverage the NPPES (National Plan & Provider Enumeration System 2019) registry to look up providers that are missing from the LEIE database, increasing their total fraudulent provider count to 12,000. They combine these fraudulent providers with a subset of 12,000 non-fraudulent providers and employ a J48 decision tree learner to classify fraud with a mean AUC of 0.96. This high AUC score is misleading because the class-balanced data set is not representative of the underlying Medicare claims data.

2.3 DoS Attack Prediction

The DoS data sets used in this study were first prepared by Calvert et al. (2018, 2019). In the first paper, Calvert et al. (2019) construct the POST data set by simulating an attack on a production server and collecting Netflow features. Class imbalance is addressed by creating 50:50 distributions with RUS, and eight learners are compared using cross-validation and the AUC metric. The results shows that all learners perform very well with AUC scores greater than 0.98. In the second paper (Calvert et al. 2018), the authors simulate Slowloris attacks and use a similar procedure to collect Netflow features. Six learners are shown to perform well and predict attack traffic with high accuracy. Given the high AUC performance of the network security data, we use these data sets as an "easier" classification task so that we can compare how imbalance affects problems with varying complexity. In our study, we combine these two POST and Slowloris data sets into one large, highly imbalanced data set.

Hasanin et al. evaluated data sampling methods by training LR, RF, and GBT learners with the Slowloris data set and then making predictions on the POST data set. In Hasanin et al. (2019), the authors use RUS to create

6 training distributions with positive class sizes ranging from 10%–60%. These training distributions are combined with feature selection methods, and the authors report that the best RUS distributions are 45:55 and 40:60. In a second paper, Hasanin et al. (2019) evaluate six data sampling strategies using the same Slowloris and POST train and test sets. The authors compare RUS and ROS to four advanced sampling techniques: SMOTE, SMOTE-borderline1, SMOTE-borderline2, and ADAptive SYNthetic. Based on AUC results, the RUS method performs better than the five alternative data methods, and the highest AUC scores are achieved when using the 90:10 class distribution. In our study, we cover a wider range of class imbalance levels by using RUS to create 17 distributions with positive class sizes ranging from 0.025%–90%. We compare the results of the easier DoS classification problem with results from two difficult Medicare problems.

3 Data Sets

This section summarizes the data sets that are used to evaluate data sampling strategies. First, we introduce the two Medicare fraud data sets that are used to evaluate ROS, RUS, and ROS-RUS. Then we introduce a network security data set that is used in an extended set of RUS experiments to test sensitivity to class imbalance and sample size. The Medicare data was first provided by Herland et al. (2018) and the network security data was collected by Calvert et al. (2019, 2018). We refer readers to these original papers for complete details on data collection and data cleaning.

3.1 CMS Medicare Data

We use two publicly available Medicare data sets: (1) Medicare Provider Utilization and Payment Data: Physician and Other Supplier (Part B) (Centers For Medicare & Medicaid Services 2018c), and (2) Medicare Provider Utilization and Payment Data: Part D Prescriber (Part D)(Centers For Medicare & Medicaid Services 2018b). Part B data contains claims for the years 2012–2016 and Part D has claims for years 2013–2017. Physicians are identified within each data set by their National Provider Identifier (NPI), a unique 10-digit identification number for healthcare providers (Centers for Medicare & Medicaid Services 2019a). Using the NPI, Herland et al. map fraud labels to the Medicare data from the LEIE repository. The LEIE is maintained by the Office of Inspector General and it lists providers that are prohibited from practicing. In addition to the provider's NPI, the LEIE also includes the reason for exclusion and a reinstatement date, if applicable. Providers that have been excluded for fraudulent activity are labelled as fraudulent within the Medicare Part B and Part D data sets.

The Part B claims data set describes the services and procedures that healthcare professionals provide to Medicare's Fee-For-Service beneficiaries. Records within the data set contain various provider-level attributes, e.g. NPI, first and last name, gender, credentials, and provider type. More importantly, records contain specific claims details that describe a provider's activity within Medicare. Examples of claims data include the procedure performed, the average charge submitted to Medicare, the average amount paid by Medicare, and the place of service. The procedures rendered are encoded using the Healthcare Common Procedures Coding System (HCPCS) (Centers For Medicare & Medicaid Services 2018a). For example, HCPCS codes 99219 and 88346 are used to bill for hospital observation care and antibody evaluation, respectively. Part B data is aggregated by: (1) provider NPI, (2) HCPCS code, and (3) place of service. The list of Part B features used for training are provided in Table 1.

Many of the Part D provider-level attributes are the same as those in the Part B data set, e.g. NPI, name, and provider type. Part D data also contains specific details about the drugs that are being prescribed to Medicare beneficiaries. Examples of prescription attributes include the drug name, the number of beneficiaries receiving the medication, the quantity being prescribed, and the cost. A full list of the Part D predictors used for classification can be found in Table 2. The Part D data was aggregated by CMS over: (1) prescriber NPI, and (2) the drug name.

3.2 Network Security Data

The network security data set used in this study is a combination of two Denial of Service (DoS) data sets, i.e. Slow HTTP POST (Calvert et al. 2019) and Slowloris (Calvert et al. 2018). Both data sets were constructed by generating attacks within a production environment and collecting Netflow features that describe the server's network traffic. The POST attack submits a POST request to the target server, specifying a large content-length within the headers. The client then sends small packets, e.g. one byte at a time, over regular intervals. The arrival of packets and the unfulfilled content-length keep the client-server connection open and tie up server resources. With enough attacks, the server runs out of resources and subsequent requests from valid clients are blocked. Similarly, the Slowloris attack exhausts server resources by sending partial messages that keep TCP connections alive. The Switchblade 4 (https://owasp.org/projects/) and a Slowloris Python script (https://github.com/gkbrk/slowloris) were used to generate attack packets for POST and Slowloris, respectively. We combined

Table 1 Description of part B features (Herland et al. 2018)

Feature	Description
Npi	Unique provider identification number
Provider_type	Medical provider's specialty (or practice)
Nppes_provider_gender	Provider's gender
Line_srvc_cnt	Number of procedures/services the provider performed
Bene_unique_cnt	Number of Medicare beneficiaries receiving the service
Bene_day_srvc_cnt	Number of Medicare beneficiary/per day services
Avg_submitted_chrg_amt	Avg. of the charges that provider submitted for service
Avg_medicare_payment_amt	Avg. payment made to a provider per claim for service
Exclusion	Fraud labels from the LEIE data set

both POST and Slowloris data sets to create a highly imbalanced data set with 3.2 million records. A description of DoS features is provided in Table 3.

3.3 Data Pre-Processing

Minimal pre-processing was performed in this study because all three data sets were curated and cleaned by the original authors. First, categorical variables were encoded using one-hot encoding. Next, train and test sets were created from each data set by randomly holding out 20% with stratified sampling. Each data set's test partition remains constant throughout all experiments, and sampling strategies are only applied to the remaining 80% of fit data. Finally, all features were normalized to continuous values in the range [0, 1]. Train and test sets were normalized by fitting a min-max scaler to the fit data and applying the scaler to the train and test sets separately.

Table 4 lists the sizes of training and test sets, the total number of predictors, and the size of the positive class for each data set. Training sets have between 2.6 and 3.7 million samples and very few positive instances. Hence, all three data sets qualify as big data problems with severe class imbalance. The Medicare data sets have between 3 and 4 positive instances for every 10,000 observations. The DoS data is less imbalanced, with roughly 20 positive instances for every 10,000 observations, but is still considered highly imbalanced. We also note that the total number of features has increased significantly during pre-processing because of the one-hot encoding of categorical variables.

4 Methodology

Data sampling methods for detecting rare classes in big data are evaluated by sampling the training set to a desired class ratio, fitting models to the sampled training data, and reporting the performance on a holdout test set. A stratified 80%/20% split is used to create the training and test sets for each data set, and the test set for each data set is constant throughout all experiments. To ensure that the test set does not influence model training, validation and hyperparameter tuning are performed using partitions within the training data. Once hyperparameters are defined, each model is fit to the training data and scored on the test set 30 times. This repetition accounts for any variation in results that may be caused by random sampling and enables statistical analysis.

This section begins by discussing the run time environment and deep learning frameworks used to carry out experiments. We then describe the DNN architectures, hyperparameters, and data sampling strategies used throughout the experiments. Finally, we discuss the performance metrics and statistical techniques that are used to evaluate results.

Table 2 Description of part D features (Herland et al. 2018)

Feature	Description
Npi	Unique provider identification number
Provider_type	Medical provider's specialty (or practice)
Bene_count	Number of distinct beneficiaries receiving drug
Total_claim_count	Number of drugs administered by provider
Total_30_day_fill_count	Number of standardized 30-day fills
Total_day_supply	Number of day's supply
Total_drug_cost	Cost paid for all associated claims
Exclusion	Fraud labels from the LEIE data set

Table 3 Description of DoS features

Feature	Description
Protocol	Transport-layer protocol of flow
Packets	Number of packets in flow
Bytes	Number of bytes in flow
Payload Rate	Non-overhead packet data per second
Flags	TCP flag fields of flow
Initial Flags	TCP flags in initial packet
Duration	Duration length (milliseconds) of flow
Payload Bytes	Size of payload (bytes)
Packets/second	Number of packets per second
Bytes/second	Number of bytes per second
Bytes/packet	Number of bytes per packet
Class	Label (Attack or Normal)

4.1 Runtime Environment

All experiments are conducted using a high-performance computing environment running Scientific Linux 7.4 (Nitrogen) (Linux 2014). Jobs are dispatched onto CPU nodes with 20 Intel(R) Xeon(R) CPU E5-2660 v3 2.60GHz processors and 128GB of RAM. Neural networks are implemented using the Keras (Chollet and et al. 2015) open-source deep learning library written in Python with its default backend, i.e. TensorFlow (Abadi et al. 2015). The specific library implementations used in this study are the default configurations of *Keras 2.1.6-tf* and *TensorFlow 1.10.0*. The scikit-learn package (Pedregosa et al. 2011) (version 0.21.1) is used for pre-processing data and establishing baseline performance results.

4.2 Baseline Models

The baseline architecture and hyperparameters used in this study were defined using a random search procedure with the Medicare Part B data set. We use 10% of the training set to validate each configuration, and we average validation results over 10 repetitions to account for any variance in the random partitions. The number of hidden layers, the number of neurons per layer, and regularization techniques are the primary focus of hyperparameter tuning in this study. We restrict experiments to deep fully-connected models, i.e. neural networks containing two or more hidden layers. We first sought a model with sufficient capacity to learn the training data and then applied regularization techniques to reduce overfitting and improve generalization to the validation set.

Table 4 Train and test set sizes

Data set		Sample Count	Feature Count	Positive Count	Positive %
Part B	Fit	3,753,896	125	1206	0.032%
	Test	938,474		302	0.032%
Part D	Fit	2,917,961	133	1028	0.035%
	Test	729,491		257	0.035%
DOS	Fit	2,621,492	77	5,317	0.203%
	Test	655,374		1,329	0.203%

We use mini-batch stochastic gradient descent (SGD) with a mini-batch size of 256. Mini-batch SGD approximates the gradient of the loss function by computing the loss over a subset of examples. This is preferred over batch gradient descent because it is computationally expensive to compute the loss over the entire data set, and increasing the number of samples that contribute to the gradient provides less than linear returns (Goodfellow et al. 2016). It has also been suggested that smaller batch sizes offer a regularization effect by introducing noise into the learning process (Wilson and Martinez 2004). We employ an advanced form of SGD that adapts parameter-specific learning rates through training, i.e. the Adam optimizer, as it has been shown to outperform other popular optimizers (Kingma and Ba 2015). The default learning rate of 0.001 is used along with default moment estimate decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The Rectified Linear Unit (ReLU) activation function is used in all hidden layer neurons, and the sigmoid activation function is used at the output layer to estimate posterior probabilities (Lippmann 1994a). The non-saturating ReLU activation function has been shown to alleviate the vanishing gradient problem and allow for faster training (Krizhevsky et al. 2012). These settings yielded the most desirable validation results during preliminary experiments.

The network topology was defined by first iteratively increasing architecture depth and width while monitoring training and validation performance. We determined that two hidden layers containing 32 neurons per layer provided sufficient capacity to overfit the model to the training data. This was indicated by near-perfect performance on the training set and an increasing error rate on the validation set. We then explored regularization techniques to eliminate overfitting and improve validation performance. One way to reduce overfitting is to reduce the total number of learnable parameters, i.e. reducing network depth or width. L^1 or L^2 regularization methods, or weight decay, add parameter penalties to the objective function that constrain the network's weights to lie within a region that is defined by a coefficient α (Goodfellow et al. 2016). Dropout simulates the ensembling of many models by randomly disabling non-output neurons with a probability $P \in [0, 1]$ during each iteration, preventing neurons from co-adapting and forcing the model to learn more robust features (Srivastava et al. 2014). Although originally designed to address internal

covariate shift and speed up training, batch normalization has also been shown to add regularizing effects to neural networks (Ioffe and Szegedy 2015). Batch normalization is similar to normalizing input data to have a fixed mean and variance, except that it normalizes the inputs to hidden layers across each batch. Through monitoring validation results, we determine that dropout with probability $P = 0.5$ combined with batch normalization is most effective. Batch normalization is applied before the activation function in each hidden unit.

Table 5 describes the baseline architecture for the Medicare Part B data set. This multilayer neural network contains two hidden layers and 5,249 trainable parameters. To determine how the number of hidden layers affects performance, we extend this model to four hidden layers following the same pattern, i.e. using 32 neurons, batch normalization, ReLU activations, and dropout in each hidden layer. We did not find it necessary to select new hyperparameters for the Medicare Part D or DoS data sets. Rather, we merely changed the size of the input layer to match the total number of features in each respective data set.

4.3 Data Sampling Strategies

Random over-sampling (ROS), random under-sampling (RUS), and hybrid ROS-RUS methods are used to vary the levels of class imbalance within each data set. Table 6 summarizes the 17 sampling strategies that are applied to the Medicare Part B and Part D data sets. The first row describes the training data prior to data sampling, and the remaining rows provide the size of the positive and negative classes after applying data sampling. n_{neg} and n_{pos} denote the total number of negative and positive samples, respectively. The level of class imbalance within each experiment's training data is represented as the ratio of total negative samples to total positive samples, i.e. $N_{neg} : N_{pos}$. These distributions cover a wide range of class imbalance, and the combination of class rarity and big data exhibited by some of these distributions poses challenges that have not been considered by related works.

Table 5 Baseline model architecture

Layer Type	# of Neurons	# of Parameters
Input	125	0
Dense	32	4032
Batch normalization	32	128
ReLU activation	32	0
Dropout $P = 0.5$	32	0
Dense	32	1056
Batch normalization	32	128
ReLU activation	32	0
Dropout $P = 0.5$	32	0
Dense	1	33
Sigmoid activation	1	0

The RUS method used in this study consists of randomly sampling from the majority class without replacement. The sampled majority class is combined with all minority samples to create the training set. One advantage of applying RUS is that the resulting training set size is reduced significantly, drastically decreasing the time required to train a model. This feature decreases turnaround time and allows for fast prototyping and hyperparameter tuning. Unfortunately, due to the extreme levels of class imbalance, very high reduction rates are required to create semi-balanced and balanced data sets. To create a class ratio of 99.9:0.1, which is still highly imbalanced, RUS-1 discards more than 2 million records from each Medicare data set. This will likely deprive the model of valuable information, potentially making RUS unsuitable for addressing imbalance in these large data sets.

The ROS method consists of duplicating minority class samples until the desired level of class imbalance is achieved. New samples introduced by ROS are exact copies from the minority class, i.e. we do not create synthetic samples. Since there are many more non-fraud cases than there are fraud, the fraud cases must be over-sampled at high rates in order to balance out the class distributions. For example, creating a 50:50 class-balanced training set from the Part B data requires sampling the minority class at a rate of 3,112%. In other words, 3,112 positive samples are created for every single positive instance, increasing the size of the minority class from 1,085 samples up to 3,377,421 and approximately doubling the size of the training data set. We note that over-sampling can also take place at the batch level during DNN training. This will help to keep memory consumption down, but it does not help to reduce training time. If batches contain half as many negative samples, then epochs will require twice as many iterations to pass over all data.

We attempt to address these concerns by using a combination of ROS and RUS, i.e. ROS-RUS, to produce three additional balanced training sets. The ROS-RUS methods begin by reducing the majority class by 90%, 75%, and 50%. Then the minority class is over-sampled until both classes are of equal size. Higher reduction rates have the advantage of decreasing the size of the training set and improving efficiency. On the other hand, lower reduction rates preserve valuable information and provide a better representation of the majority class. Unlike plain RUS, the ROS-RUS method allows us to keep a greater percentage of the majority class, reducing the risk of discarding too many negative samples and under-representing the majority

Table 6 Sampling strategies and distributions

Method	$n_{neg} : n_{pos}$	CMS Part B		CMS Part D	
		n_{neg}	n_{pos}	n_{neg}	n_{pos}
Baseline	99.97:0.03	3,377,421	1,085	2,916,933	1,028
RUS-1	99.9:0.1	773,092	1,085	1,027,052	1,028
RUS-2	99.5:0.5	194,202	1,085	204,477	1,028
RUS-3	99:1	107,402	1,085	101,801	1,028
RUS-4	80:20	4,390	1,085	4,084	1,028
RUS-5	60:40	1,620	1,085	1,546	1,028
RUS-6	50:50	1,085	1,085	1,028	1,028
RUS-7	40:60	710	1,085	671	1,028
ROS-1	99.9:0.1	3,377,421	3,385	2,916,933	2,920
ROS-2	99.5:0.5	3,377,421	16,969	2,916,933	14,659
ROS-3	99:1	3,377,421	33,635	2,916,933	29,401
ROS-4	80:20	3,377,421	844,130	2,916,933	729,263
ROS-5	60:40	3,377,421	2,251,375	2,916,933	1,944,626
ROS-6	50:50	3,377,421	3,377,421	2,916,933	2,916,929
ROS-7	40:60	3,377,421	5,064,780	2,916,933	4,375,404
ROS-RUS-1	50:50	1,688,710	1,688,710	1,458,466	1,458,466
ROS-RUS-2	50:50	844,355	844,355	729,233	729,233
ROS-RUS-3	50:50	337,742	337,742	291,693	291,693

class. Since the majority group has been decreased in size through RUS, the over-sampling rate required to balance the classes is going to be less than would be required if using plain ROS. As shown in Table 6, the largest ROS-RUS distribution has fewer observations than the original data set. We find these methods most favorable when working with big data and class rarity, as they simultaneously maximize efficiency and performance.

4.4 Performance Evaluation

The confusion matrix (Table 7) is constructed by comparing predicted labels to ground truth labels, where the predicted labels are dependent on output scores and a decision threshold. We found that the default decision threshold performs very poorly when training data is imbalanced, and it caused most models to predict all test observations to be from the negative class. Therefore, we rely on the ROC AUC to compare model performance because it does not depend on the threshold and it is robust to class imbalance. The ROC curve plots the False Positive Rate (FPR) against the True Positive Rate (TPR) and the AUC summarizes the curve with a score between 0 and 1. Given a high AUC score, we can then identify the optimum operating point on the ROC curve to maximize the TPR.

$$G\text{-}Mean = \sqrt{TPR \times TNR} \tag{2}$$

To make predictions on the test set, we use a procedure (Section 4.5) to find the optimal classification thresholds that maximize the TPR and the G-Mean. The G-Mean (Eq. 2) summarizes the model's total predictive power by combining the TPR with the True Negative Rate (TNR). We present the average G-Mean results for the Medicare Part B and Part D data sets in Section 5.

4.5 Threshold Moving

Selecting an appropriate classification threshold is crucial to maximizing class-wise performance, and the default threshold of 0.5 is rarely optimal when neural networks are trained with imbalanced data. Since threshold moving trades true positives for true negatives, the optimal threshold should be driven by application requirements and will vary from task to task. In this study, we prefer a high TPR over a high TNR, because we are most interested in detecting fraud and network attacks. Furthermore, we would like to approximately balance the TPR and TNR to avoid many false positives. Following this strategy, we defined a threshold selection procedure that iterates over all possible thresholds and selects the one that maximizes the G-Mean under the constraint that $TPR > TNR$. This procedure is applied during each distribution's 10 validation runs, and the average threshold is used to make predictions on the test set.

4.6 Significance Testing

Tukey's HSD test ($\alpha = 0.05$) is used to estimate the significance of our results and identify the best performing methods. Tukey's HSD test is a multiple comparison procedure that determines which method means are statistically different from each other by identifying differences that are greater than the expected standard error. Methods are assigned to alphabetic groups based on the statistical difference of AUC means, e.g. group *a* is significantly different from group *b*. In other words, HSD results allow us to assert with 95% confidence that differences in AUC scores did not occur by chance.

5 Results

This section discusses the data sampling results that were obtained using three severely class-imbalanced data sets, a wide range of class imbalance levels, and two DNN architectures. AUC significance testing, training time analysis, and G-Mean performance are used to select the best sampling strategies for detecting rare classes in big data. We begin by presenting several baseline performance benchmarks, and then proceed to compare ROS and RUS methods. We then combine these methods to show how a hybrid ROS-RUS method is able to maximize both performance and efficiency. Finally, we illustrate the importance of thresholding by comparing G-Mean results on Medicare data sets.

5.1 Baseline Model Performance

The baseline AUC results that were obtained using the original imbalanced Part B and Part D distributions are listed in Table 8. The two-hidden-layer network (DNN-2) and four-hidden-layer network (DNN-4) results will be used as a reference to evaluate the efficacy of data sampling methods throughout the remainder of this study. Results for three traditional machine learning algorithms have been included to validate our selection of DNN hyperparameters.

All learners perform similarly on the Part B data set, and the logistic regression learner performs the best with an average AUC of 0.8076. DNN-2 performed best on the Part D data set with an AUC of 0.7663. Lower AUC scores and an increase in variance between learners leads us to believe that Part D fraud classification is more difficult than Part B fraud classification. For both data sets, performance degrades as the number of hidden layers increases from two to four. We expect results to improve as class imbalance is addressed through data sampling, and upcoming sections will use a wide range of data sampling strategies to show how varying class imbalance levels affects performance.

Table 7 Confusion matrix

	Actual positive	Actual negative
Predicted positive	True Positive (TP)	False Positive (FP)
Predicted negative	False Negative (FN)	True Negative (TN)

Table 8 Baseline AUC results

Data set	CMS Part B	CMS Part D
DNN-2	0.8058	0.7663
DNN-4	0.8018	0.7346
Logistic regression	0.8076	0.7299
Random forest	0.7937	0.7209
Gradient boosted trees	0.7990	0.7118

5.2 RUS Results

RUS results for the Medicare Part B and Part D data sets are illustrated with boxplots in Figs. 1 and 2. RUS-2 and RUS-3 are the best performing under-sampling methods for both Medicare data sets and both methods succeed in outperforming the baseline models. Surprisingly, these methods have training distributions that are still highly imbalanced, i.e. positive class sizes of 0.5% and 1%. As we begin to reduce the class imbalance in the training sets of RUS-1, RUS-2, and RUS-3, models show gradual improvements on AUC performance. When we attempt to decrease class imbalance any further, e.g. RUS-4, RUS-5, etc., performance begins to deteriorate until eventually AUC scores fall below the baseline threshold. In this scenario, RUS does not work well when the majority class is under-sampled to a 50:50 distribution.

We believe that the peak in RUS performance at RUS-3 is caused by a trade-off between class imbalance and information loss. The size of the majority class can only be reduced by so much before it becomes underrepresented and model performance begins to suffer. Results also indicate an increase in AUC variance as smaller samples are taken. We attribute this increase in variance to "lucky" and "unlucky" draws from the negative class as the sample size decreases.

The combination of high imbalance and big data prevents us from fully addressing class imbalance with RUS. When the negative class is very large and there are few positive instances, then most of the negative class must be discarded in order to establish balance and we risk misrepresenting the negative class. For example, RUS-3 requires just 3% of the majority class to create a training set with a 99:1 class distribution. In other words, 97% of the majority class,

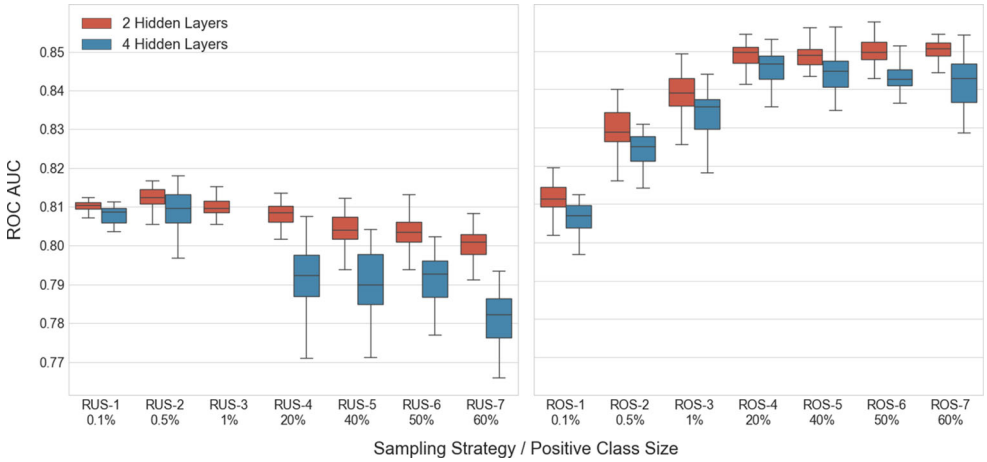

Fig. 1 CMS Part B: ROS vs RUS

or 3.5 million observations, is being discarded. The fact that RUS-6 has a class-balanced training set and performs worse than baseline models suggests that determining an effective sample size is more important than addressing class imbalance. We believe that this matter of determining effective negative sample sizes is crucial to modeling rare cases in the presence of big data.

To better understand the effects of RUS, class imbalance, and insufficient sample sizes, Fig. 3 plots AUC scores for two-layer networks over a wider range of class imbalance levels and incorporates a third data set. The DoS security data does exhibit both class rarity and big data characteristics, but it is a much easier classification task. The underlying tradeoff between class imbalance and information loss exists in all three data sets, but the Part D data set is clearly the most sensitive. Part D AUC scores fluctuate from below 0.70 to above 0.80 as the positive class size varies, while the Part B and DoS performance remains relatively stable between 0.25% and 90.0%. Part B results show a slow upward trend as the size of the positive class grows, but performance begins to suffer once the positive class grows beyond 1%. The DoS performance decreases when the positive class size falls below 0.025% or above 90.0%, but AUC performance is mostly stable and never falls below 0.988. These results suggest that each data set is uniquely sensitive to class imbalance and sample size.

5.3 ROS Performance

Most of the ROS methods presented in Figs. 1 and 2 outperform all RUS and baseline results. For both data sets, increasing the size of the positive class to just 0.1% (ROS-1) is all that is required to match the best RUS performance. This shows that even modest over-sampling can improve overall model performance. As the size of the positive class increases and the level of class imbalance decreases, we see a steady increase in average AUC scores. This increase in performance plateaus around ROS-4, when the positive class size reaches 20% of the training set. ROS-4, ROS-5, ROS-6, and ROS-7 are the best performing over-sampling methods for both data sets, indicating that ROS generally performs best when class imbalance is eliminated from the training set.

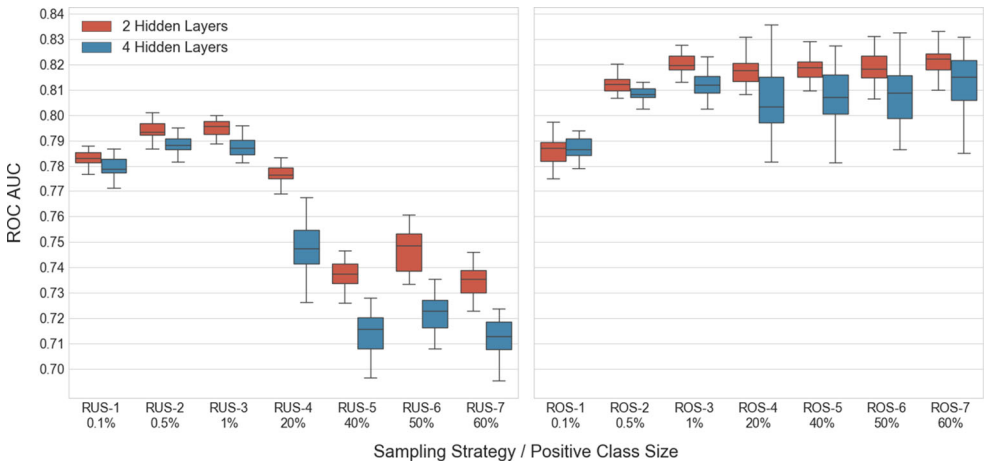

Fig. 2 CMS Part D: ROS vs RUS

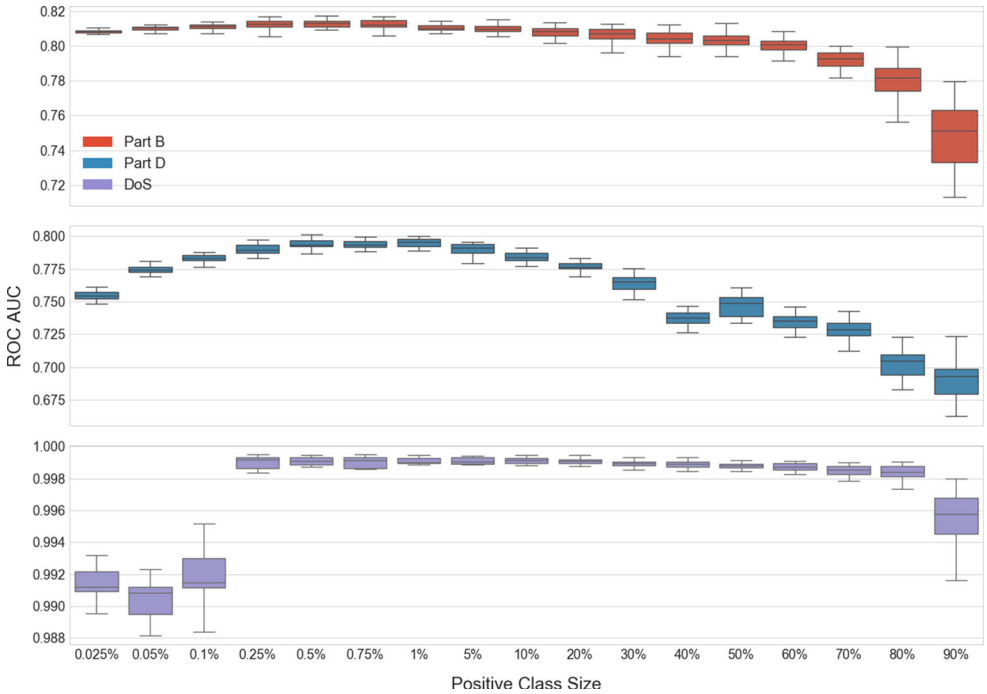

Fig. 3 RUS results: Class imbalance vs insufficient sample sizes

These results are in correspondence with those presented by Buda et al. (2018), suggesting that over-sampling be our go-to method for training deep models with class-imbalanced data. While we agree that ROS should work well for most problems, we argue that there are non-negligible costs to consider when using ROS to address class rarity and big data. Given that the negative class is so large, e.g. millions of samples, and there are so few positive observations, very high over-sampling rates are required to balance out class distributions. For example, Part B's ROS-4 method must up-sample the positive class by 778% in order to obtain a training set with an 80:20 distribution. Some will argue these costs can be avoided by over-sampling at the batch level during training, but this will only reduce the memory footprint. If over-sampling at the batch level, additional iterations will be required to allow each epoch to see all samples from the majority class. This adds significant overhead to model training and, depending on available resources, may not always be feasible.

5.4 ROS-RUS Hybrid Performance

The hybrid ROS-RUS method results are listed in Table 9. All three hybrid ROS-RUS methods perform exceptionally well on both Medicare data sets, outperforming all RUS and baseline models. For brevity, we did not include results from four-hidden-layer networks, noting that the two-hidden-layer networks continue to outperform respective four-layer networks.

ROS-RUS-2 performs the best with a mean AUC of 0.8509 and 0.8234 on the Part B and Part D data sets, respectively. This hybrid method reduces the size of the negative class by 75% and then over-samples the positive class until balance is achieved. The fact that ROS-RUS distributions have smaller training sets than ROS distributions indirectly confirms that high ROS performance is not due to an increased size in training sets. Even ROS-RUS-3, which reduces the size of the negative class by 90%, performs exceptionally well with mean AUC scores of 0.8477 and 0.8225. The ability to drastically reduce the size of the training set and still maximize performance is ideal for addressing severe class imbalance within big data problems.

5.5 Significance Testing

Tukey's HSD test ($\alpha = 0.05$) was used to examine the differences between AUC scores. Table 9 presents these results using the compact letter display. Methods are listed in descending order by their average AUC. Four-layer network results have been excluded for brevity, but their results are similar to those produced by two-layer networks.

The hybrid ROS-RUS methods perform the best on both Medicare data sets. The overlapping letters indicate that the differences in mean AUC scores are not significant. While their AUC scores are more or less equal, we prefer ROS-RUS-2 or ROS-RUS-3 over ROS-RUS-1 because they are more efficient. ROS-RUS-2 and ROS-RUS-3 manage to discard 75% and 90% of the negative class, respectively. As

Table 9 Tukey's HSD Results (2 Hidden Layers)

CMS Part B			CMS Part D		
Sampling strategy	HSD group	Mean AUC	Sampling strategy	HSD group	Mean AUC
ROS-RUS-2	a	0.8509	ROS-RUS-2	a	0.8234
ROS-6	a	0.8504	ROS-RUS-1	ab	0.8230
ROS-7	a	0.8503	ROS-RUS-3	ab	0.8225
ROS-RUS-1	a	0.8500	ROS-7	abc	0.8218
ROS-4	a	0.8484	ROS-3	abc	0.8202
ROS-RUS-3	a	0.8477	ROS-5	abc	0.8189
ROS-5	a	0.8454	ROS-6	bc	0.8181
ROS-3	b	0.8382	ROS-4	c	0.8173
ROS-2	c	0.8280	ROS-2	d	0.8122
RUS-3	d	0.8124	RUS-3	e	0.7951
RUS-2	d	0.8122	RUS-2	e	0.7943
ROS-1	de	0.8114	ROS-1	f	0.7863
RUS-1	de	0.8102	RUS-1	f	0.7830
RUS-4	def	0.8076	RUS-4	g	0.7756
Baseline	ef	0.8058	Baseline	h	0.7663
RUS-5	fg	0.8043	RUS-6	i	0.7465
RUS-6	fg	0.8027	RUS-5	j	0.7375
RUS-7	h	0.7994	RUS-7	j	0.7349

mentioned earlier, these substantial reductions in training set sizes are a major advantage for big data problems.

Summarizing the results, all ROS methods that increase the size of the positive class to 20% or higher perform well and yield HSD groups that contain the letter *a*. The only exception to this is ROS-6 on the Part D data set, which falls into group *bc*. ROS-1 and ROS-2 perform significantly worse than ROS-3, but still outperform the baseline model.

The only two RUS methods that perform significantly better than the baseline model on both data sets are RUS-2 and RUS-3. RUS-1 and RUS-4 achieve higher average AUC scores when compared to the baseline, but these results are only considered significant on the Part D data set. We attribute the poor RUS performance to information loss, and suggest that future works explore methods for determining effective sample sizes.

5.6 Training Time Analysis

Often times, the process of training and deploying machine learning models is constrained by non-functional requirements. When working with large data sets, one requirement may be to minimize the total training time. We take this into consideration and present the training times of the best methods from each group of data sampling strategies in Table 10. By taking both efficiency and performance into consideration, we are better able to select the best data sampling strategy for big data tasks. Since the total number of training epochs is constant between methods, we compare methods using the average time to train a single epoch.

For the Part B data set, the baseline model takes an average of 63.0775 seconds to train a single epoch containing 3.4 million records. We are mostly concerned with ROS-RUS-2 and ROS-6, as they achieve the highest AUC scores on the Part B data sets. ROS-6, with an average training time of 128.2847 seconds, is roughly 2× slower than the baseline and 4× slower than ROS-RUS-2. ROS-RUS-2, on the other hand, is roughly 2× faster than the baseline model. In this scenario, we prefer ROS-RUS-2 as it maximizes both AUC performance and efficiency. We point out that RUS-3 also performed significantly better than the baseline model and only required an average of 1.9213 seconds per training epoch. With a 30× speedup in training, RUS-3 appears to be a great strategy for preliminary experiments and rapid prototyping. Since training times are proportional to the size of the training set, the results on the Part D data set follow a similar pattern.

5.7 Thresholding Results

Optimal thresholds were calculated during model validation using the procedure defined in Section 4.5. Preliminary results showed that optimal thresholds are approximately equal to the relative size of the positive class. For example, the Part B data set has a relative minority class size of

Table 10 Average time per training epoch (2 Hidden Layers)

CMS Part B			CMS Part D		
Sampling strategy	Time (s)	# Train samples	Sampling strategy	Time (s)	# Train samples
RUS-3	1.9213	108,487	RUS-3	1.4197	102,829
ROS-RUS-2	31.0784	1,677,710	ROS-RUS-2	17.3680	1,458,466
Baseline	63.0775	3,378,506	Baseline	25.5816	2,917,961
ROS-6	128.2847	6,754,842	ROS-7	82.7704	7,292,337

0.0003, and the average optimal threshold was calculated to be 0.0002. Similarly, RUS-6 produced a class-balanced distribution from the Part B data and the average optimal threshold is approximately equal to the default threshold, i.e. 0.4970. This relationship between the minority class size and the optimal threshold is due to neural networks estimating Bayesian posterior probabilities (Lippmann 1994b). In general, the default threshold of 0.5 is never optimal when neural networks are trained with imbalanced data.

Part B thresholding results are grouped by the number of hidden layers and optimal thresholds are plotted against the relative size of the positive class in Fig. 4. Results indicate a strong linear relationship with r^2 = 0.987 and p-value = 6.70e-132 for the two-hidden-layer networks. There is also a clear difference between thresholds generated with two hidden layers and thresholds generated with four hidden layers. Therefore, we suggest that thresholds always be optimized when training neural networks with imbalanced data.

5.8 Class-Wise Performance

We summarize the performance of the positive and negative class using optimal G-Mean scores, i.e. scores calculated using the optimal classification thresholds that were identified during model validation. We then compare these optimal scores to the default G-Mean scores that are calculated using the default threshold of 0.5. The optimal thresholds and G-Mean scores for the Medicare Part B and Part D data sets are listed in Table 11. The maximum score for each distribution and data set is listed in bold font.

We found that G-Mean results correspond closely to AUC results when the optimal threshold is used. For example, RUS-2 and RUS-3 are the best performing under-sampling methods based on AUC criteria. Similarly, RUS-2 and RUS-3 receive the highest G-Mean scores when the optimal threshold is used. The same can be said for the ROS-6 and ROS-7 results on the Medicare Part B data set. This intuitively makes sense, as a maximum AUC implies the existence of a threshold that will maximize TPR and TNR.

The correspondence between AUC and class-wise performance does not exist, however, when the default classification threshold is used. When the default threshold of 0.5 is used, RUS-2 and RUS-3 predict all records to be negative, yielding G-Mean scores of 0.0. In fact, if we use any performance metric that is dependent on a classification threshold and we do not optimize the threshold, then our perception of the best sampling strategy will change drastically. When we judge model performance by the G-Mean score using the default threshold, then RUS-5 and RUS-6 become the preferred models. We would conclude that RUS works best when the majority is under-sampled to the point of class imbalance. Of course, this would be an oversight on our part and model performance would suffer because of it. This false perception can be avoided by using threshold-agnostic performance metrics or by optimizing decision thresholds whenever training data is imbalanced.

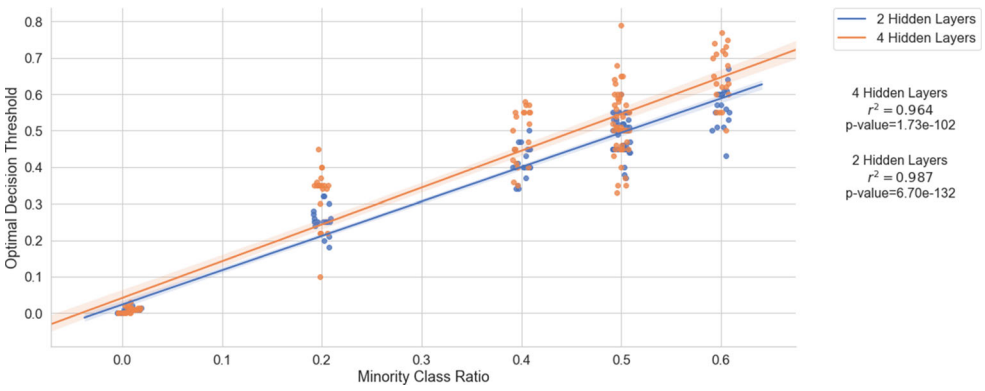

Fig. 4 Part B: Positive class size vs optimal threshold

Table 11 Optimal thresholds and average G-Means

Sampling strategy	Positive class size	CMS Part B			CMS Part D		
		Optimal Threshold	Default G-Mean	Optimal G-Mean	Optimal Threshold	Default G-Mean	Optimal G-Mean
RUS-1	0.1%	0.0009	0.0000	**0.7243**	0.0007	0.0000	**0.7093**
RUS-2	0.5%	0.0064	0.0000	**0.7402**	0.0052	0.0000	**0.7281**
RUS-3	1%	0.0110	0.0000	**0.7383**	0.0118	0.0000	**0.7325**
RUS-4	20%	0.2680	0.1421	**0.7338**	0.2257	0.2373	**0.7086**
RUS-5	40%	0.4200	**0.7320**	0.7212	0.4348	0.6747	**0.6832**
RUS-6	50%	0.4970	0.7190	**0.7195**	0.5008	0.6898	**0.6899**
RUS-7	60%	0.5730	0.6670	**0.7154**	0.5986	0.6546	**0.6814**
ROS-1	0.1%	0.0007	0.0000	**0.7193**	0.0006	0.0000	**0.6998**
ROS-2	0.5%	0.0062	0.0000	**0.7501**	0.0058	0.0029	**0.7421**
ROS-3	1%	0.0110	0.0000	**0.7338**	0.0126	0.0125	**0.7456**
ROS-4	20%	0.2410	0.6268	**0.7549**	0.2286	**0.7517**	0.7428
ROS-5	40%	0.4080	**0.7629**	0.7582	0.3803	0.7371	**0.7459**
ROS-6	50%	0.4530	0.7657	**0.7692**	0.3864	**0.7464**	0.7332
ROS-7	60%	0.5630	0.7560	**0.7701**	0.5449	0.7438	**0.7483**

6 Conclusion

This study evaluates the use of data sampling methods for training artificial neural networks with highly imbalanced big data. The Medicare Part B and Part D data sets used in this study have 4.6 and 3.6 million records, respectively, with positive class sizes $<$ 0.035%. The DoS data set contains over 3.2 million network samples and has a positive class size of just 0.20%. ROS, RUS, and ROS-RUS are used to create 17 new training distributions with varying levels of class imbalance, i.e. positive class sizes between 0.025%–90%. Experiments are repeated 30 times using two DNN architectures and statistical analysis is used to identify the best methods based on average AUC scores.

Similar to related works, ROS methods perform significantly better than all RUS and baseline methods across both Medicare data sets. The ROS performance came at a cost, however, as the large over-sampling rates that were required to address the combination of big data and high imbalance increased training times. Addressing high imbalance within big data using RUS methods proved difficult, as AUC results suggest that each data set is uniquely sensitive to class imbalance and sample size. For example, Part B AUC results improve when RUS is used to reduce the imbalance down to 99:1, but further reducing the level of imbalance misrepresents the majority class and degrades performance. The easier classification problem, i.e. the DoS data set, maintains a relatively stable AUC score above 0.988 and does not appear to be affected by class imbalance or an underrepresented majority class. The ROS-RUS hybrid methods ended up performing as good as, or better, than the ROS methods based on AUC scores. In addition to maximizing AUC performance, the ROS-RUS methods discard up to 90% of the majority class and drastically reduce training costs. Since ROS-RUS performs statistically the same as ROS, and the ROS-RUS distributions contain fewer training samples, we can conclude that ROS outperforming RUS is not due to the larger training sets of ROS. We found thresholding to be a critical component of training neural networks with imbalanced data, and a comparison of G-Mean scores shows that using default thresholds may lead to poor model selection. Given the nature of big data, functional requirements, and the results of this study, we recommend the use of hybrid ROS-RUS methods for treating high imbalance in big data problems.

Future works should evaluate these plain data sampling strategies across a wider range of domains and network architectures, e.g. natural language processing. Additionally, more intelligent data sampling techniques can be incorporated into these experiments, especially editing techniques that remove redundant samples. Finally, methods for predicting effective sample sizes should be explored and integrated into RUS methods in order to reduce the risk of misrepresenting the majority class.

Acknowledgements The authors would like to thank the various members of the Data Mining and Machine Learning Laboratory at Florida Atlantic University for assistance with reviews.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. http://tensorflow.org/.

Ahmed, S.E. (2014). *Perspectives on big data analysis: methodologies and applications*. USA: Amer Mathematical Society.

Anand, R., Mehrotra, K.G., Mohan, C.K., Ranka, S. (1993). An improved algorithm for neural network classification of imbalanced training sets. *IEEE Transactions on Neural Networks*, *4*(6), 962–969. https://doi.org/10.1109/72.286891.

Bauder, R.A., & Khoshgoftaar, T.M. (2016). A novel method for fraudulent medicare claims detection from expected payment deviations (application paper). In *2016 IEEE 17Th international conference on information reuse and integration (IRI)* (pp. 11–19). https://doi.org/10.1109/IRI.2016.11.

Bauder, R.A., & Khoshgoftaar, T.M. (2016). A probabilistic programming approach for outlier detection in healthcare claims. In *2016 15Th IEEE international conference on machine learning and applications (ICMLA), pp. 347–354*, https://doi.org/10.1109/ICMLA.2016.0063.

Bauder, R.A., & Khoshgoftaar, T.M. (2018). The detection of medicare fraud using machine learning methods with excluded provider labels. In *FLAIRS conference*.

Bauder, R.A., Khoshgoftaar, T.M., Hasanin, T. (2018). An empirical study on class rarity in big data. In *2018 17Th IEEE international conference on machine learning and applications (ICMLA)* (pp. 785–790). https://doi.org/10.1109/ICMLA.2018.00125.

Bauder, R.A., Khoshgoftaar, T.M., Richter, A., Herland, M. (2016). Predicting medical provider specialties to detect anomalous insurance claims. In *2016 IEEE 28Th international conference on tools with artificial intelligence (ICTAI)* (pp. 784–790). https://doi.org/10.1109/ICTAI.2016.0123.

Branting, L.K., Reeder, F., Gold, J., Champney, T. (2016). Graph analytics for healthcare fraud risk estimation. In *2016 IEEE/ACM International conference on advances in social networks analysis and mining (ASONAM), pp. 845–851*. https://doi.org/10.1109/ASONAM.2016.7752336.

Buda, M., Maki, A., Mazurowski, M.A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259. https://doi.org/10.1016/j.neunet.2018.07.011. http://www.sciencedirect.com/science/article/pii/S0893608018302107.

Calvert, C., Kemp, C., Khoshgoftaar, T.M., Najafabadi, M.M. (2018). Detecting of slowloris attacks using netflow traffic. In *24Th ISSAT international conference on reliability and quality in design* (pp. 191–6).

Calvert, C., Kemp, C., Khoshgoftaar, T.M., Najafabadi, M.M. (2019). Detecting slow http post dos attacks using netflow features. In *FLAIRS conference*.

Centers For Medicare & Medicaid Services (2018). Hcpcs general information. https://www.cms.gov/Medicare/Coding/MedHCPCSGenInfo/index.html.

Centers For Medicare & Medicaid Services (2018). Medicare provider utilization and payment data: Part d prescriber. https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html.

Centers For Medicare & Medicaid Services (2018). Medicare provider utilization and payment data: Physician and other supplier. https://www.cms.gov/research-statistics-data-and-systems/statistics-trends-and-reports/medicare-provider-charge-data/physician-and-other-supplier.html.

Centers for Medicare & Medicaid Services (2019). National provider identifier standard (npi). https://www.cms.gov/Regulations-and-Guidance/Administrative-Simplification/NationalProvIdentStand/.

Centers for Medicare & Medicaid Services (2019). Physician compare datasets. https://data.medicare.gov/data/physician-compare.

Chahal, K., Grover, M., Dey, K., Shah, R.R. (2019). A hitchhiker's guide on distributed training of deep neural networks. Journal of Parallel and Distributed Computing. https://doi.org/10.1016/j.jpdc.2019.10.004.

Chandola, V., Sukumar, S.R., Schryver, J.C. (2013). Knowledge discovery from massive healthcare claims data. In *KDD*.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P. (2002). Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, *16*(1), 321–357. http://dl.acm.org/citation.cfm?id=1622407.1622416.

Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W. (2003). Smoteboost: Improving prediction of the minority class in boosting. In Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (Eds.) *Knowledge discovery in databases: PKDD 2003* (pp. 107–119). Berlin: Springer.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning.

Chollet, F., et al. (2015). Keras. https://keras.io.

Dean, J., & Ghemawat, S. (2008). Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113. https://doi.org/10.1145/1327452.1327492.

Dumbill, E. (2012). What is big data? : an introduction to the big data landscape. http://radar.oreilly.com/2012/01/what-is-big-data.html.

Feldman, K., & Chawla, N.V. (2015). Does medical school training relate to practice? evidence from big data. In *Big data*.

Fernández, A., del Río, S., Chawla, N.V., Herrera, F. (2017). An insight into imbalanced big data classification: outcomes and challenges. *Complex & Intelligent Systems*, *3*(2), 105–120. https://doi.org/10.1007/s40747-017-0037-9.

Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep learning*. Cambridge: The MIT Press.

Han, H., Wang, W.Y., Mao, B.H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In Huang, D.S., Zhang, X.P., Huang, G.B. (Eds.) *Advances in intelligent computing* (pp. 878–887). Berlin: Springer.

Hasanin, T., Khoshgoftaar, T.M., Leevy, J.L., Bauder, R.A. (2019). Severely imbalanced big data challenges: investigating data sampling approaches. *Journal of Big Data*, *6*(1), 107. https://doi.org/10.1186/s40537-019-0274-4.

Hasanin, T., Khoshgoftaar, T.M., Leevy, J.L., Seliya, N. (2019). Examining characteristics of predictive models with imbalanced big data. *Journal of Big Data*, *6*(1), 69. https://doi.org/10.1186/s40537-019-0231-2.

He, H., & Garcia, E.A. (2009). Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, *21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239.

Herland, M., Bauder, R.A., Khoshgoftaar, T.M. (2017). Medical provider specialty predictions for the detection of anomalous medicare insurance claims. In *2017 IEEE International conference on information reuse and integration (IRI)* (pp. 579–588), https://doi.org/10.1109/IRI.2017.29.

Herland, M., Bauder, R.A., Khoshgoftaar, T.M. (2019). The effects of class rarity on the evaluation of supervised healthcare fraud detection models. *Journal of Big Data*, *6*(1), 21. https://doi.org/10.1186/s40537-019-0181-8.

Herland, M., Khoshgoftaar, T.M., Bauder, R.A. (2018). Big data fraud detection using multiple medicare data sources. *Journal of Big Data*, *5*(1), 29. https://doi.org/10.1186/s40537-018-0138-3.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd international conference on international conference on machine learning*, (Vol. 37 pp. 448–456): ICML'15.

Jo, T., & Japkowicz, N. (2004). Class imbalances versus small disjuncts. *SIGKDD Explor. Newsl.*, *6*(1), 40–49. https://doi.org/10.1145/1007730.1007737.

Johnson, J.M., & Khoshgoftaar, T.M. (2019). Deep learning and data sampling with imbalanced big data. In *2019 IEEE 20Th international conference on information reuse and integration for data science (IRI)* (pp. 175–183). https://doi.org/10.1109/IRI.2019.00038.

Johnson, J.M., & Khoshgoftaar, T.M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, *6*(1), 27. https://doi.org/10.1186/s40537-019-0192-5.

Kankanhalli, A., Hahn, J., Tan, S., Gao, G. (2016). Big data and analytics in healthcare: Introduction to the special section. *Information Systems Frontiers*, *18*(2), 233–235. https://doi.org/10.1007/s10796-016-9641-2.

Kennedy, R.K.L., Khoshgoftaar, T.M., Villanustre, F., Humphrey, T. (2019). A parallel and distributed stochastic gradient descent implementation using commodity clusters. *Journal of Big Data*, *6*(1), 16. https://doi.org/10.1186/s40537-019-0179-2.

Khoshgoftaar, T.M., Gao, K., Napolitano, A., Wald, R. (2014). A comparative study of iterative and non-iterative feature selection techniques for software defect prediction. *Information Systems Frontiers*, *16*(5), 801–822. https://doi.org/10.1007/s10796-013-9430-0.

Kingma, D.P., & Ba, J. (2015). Adam: a method for stochastic optimization. arXiv:abs/1412.6980.

Ko, J., Chalfin, H., Trock, B., Feng, Z., Humphreys, E., Park, S.W., Carter, B., D Frick, K., Han, M. (2015). Variability in medicare utilization and payment among urologists. Urology 85. https://doi.org/10.1016/j.urology.2014.11.054.

Krizhevsky, A., Nair, V., Hinton, G. Cifar-10 (canadian institute for advanced research) http://www.cs.toronto.edu/kriz/cifar.html.

Krizhevsky, A., Sutskever, I., Hinton, E.G. (2012). Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems, 25. https://doi.org/10.1145/3065386.

Kubat, M., Holte, R.C., Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, *30*(2), 195–215. https://doi.org/10.1023/A:1007452223027.

LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, *52*, 436 EP. https://doi.org/10.1038/nature14539.

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/. Accessed: 2018-11-15.

Lee, H., Park, M., Kim, J. (2016). Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In *2016 IEEE International conference on image processing (ICIP)* (pp. 3713–3717). https://doi.org/10.1109/ICIP.2016.7533053.

Leevy, J.L., Khoshgoftaar, T.M., Bauder, R.A., Seliya, N. (2018). A survey on addressing high-class imbalance in big data. *Journal of Big Data*, *5*(1), 42. https://doi.org/10.1186/s40537-018-0151-6.

Ling, C.X., & Sheng, V.S. (2007). Cost-sensitive Learning and the Class Imbalanced Problem.

Linux, S. (2014). About. https://www.scientificlinux.org/about/.

Lippmann, R.P. (1994). Neural networks, bayesian a posteriori probabilities, and pattern classification. In Cherkassky, V., Friedman, J.H., Wechsler, H. (Eds.) *From statistics to neural networks* (pp. 83-104). Berlin: Springer.

Lippmann, R.P. (1994). Neural networks, bayesian a posteriori probabilities, and pattern classification. In Cherkassky, V., Friedman, J.H., Wechsler, H. (Eds.) *From statistics to neural networks* (pp. 83-104). Berlin: Springer.

Liu, X., Wu, J., Zhou, Z. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550. https://doi.org/10.1109/TSMCB.2008.2007853.

Masko, D., & Hensman, P. (2015). The impact of imbalanced training data for convolutional neural networks. KTH, School of Computer Science and Communication (CSC).

National Plan & Provider Enumeration System (2019). Nppes npi registry. https://npiregistry.cms.hhs.gov/registry/.

Office of Inspector General (2019). Leie downloadable databases. https://oig.hhs.gov/exclusions/exclusions_list.asp.

Orenstein, E.C., Beijbom, O., Peacock, E.E., Sosik, H.M. (2015). Whoi-plankton- a large scale fine grained visual recognition benchmark dataset for plankton classification. arXiv:abs/1510.00745.

OWASP: Owasp http post tool. https://www.owasp.org/index.php/OWASP_HTTP_Post_Tool.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

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

Provost, F., & Fawcett, T. (1999). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 43–48).

Rao, R.B., Krishnan, S., Niculescu, R.S. (2006). Data mining for improved cardiac care. *SIGKDD Explor. Newsl.*, *8*(1), 3–10. https://doi.org/10.1145/1147234.1147236.

Requeno, J., Merseguer, J., Bernardi, S., Perez-Palacin, D., Giotis, G., Papanikolaou, V. (2019). Quantitative analysis of apache storm applications: the newsasset case study. *Information Systems Frontiers*, *21*(1), 67–85. https://doi.org/10.1007/s10796-018-9851-x.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958. http://dl.acm.org/citation.cfm?id=2627435.2670313.

Sun, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. Ph.D. thesis, Waterloo, Ont., Canada, Canada. AAINR34548.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. arXiv:abs/1605.02688.

Tukey, J.W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, *5*(2), 99–114. http://www.jstor.org/stable/3001913.

U.S. Government, U.S. Centers for Medicare & Medicaid Services: The official u.s. government site for medicare. https://www.medicare.gov/.

Wei, W., Li, J., Cao, L., Ou, Y., Chen, J. (2013). Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, *16*(4), 449–475. https://doi.org/10.1007/s11280-012-0178-0.

Weiss, G.M. (2004). Mining with rarity: A unifying framework. *SIGKDD Explor. Newsl.*, *6*(1), 7–19. https://doi.org/10.1145/1007730.1007734.

Wilson, D., & Martinez, T. (2004). The general inefficiency of batch training for gradient descent learning. *Neural networks :, the official journal of the International Neural Network Society*, *16*, 1429–51. https://doi.org/10.1016/S0893-6080(03)00138-2.

Wilson, D.L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-2*(3), 408–421. https://doi.org/10.1109/TSMC.1972.4309137.

Witten, I.H., Frank, E., Hall, M.A., Pal, C.J. (2016). *Data mining, fourth edition: practical machine learning tools and techniques*, 4th edn. San Francisco: Morgan Kaufmann Publishers Inc.

Yaltirakli, G. Slowloris. https://github.com/gkbrk/slowloris.

Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10* (pp. 10–10). Berkeley: USENIX Association. http://dl.acm.org/citation.cfm?id=1863103.1863113.

Justin M. Johnson is a PhD candidate at Florida Atlantic University. He has a background in Software Engineering, and his research interests include data mining, machine learning, big data analytics, and deep learning. Justin is first author of four refereed journal and conference papers, and he was awarded for best paper at the IEEE International Conference on Machine Learning and Applications (ICMLA 2019).

Dr. Taghi M. Khoshgoftaar is the Motorola Endowed Chair professor of the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University and the Director of NSF Big Data Training and Research Laboratory. His research interests are in big data analytics, data mining and machine learning, health informatics and bioinformatics, social network mining, and software engineering. He has published more than 750 refereed journal and conference papers in these areas. He is the Co-Editor-in Chief of the Journal of Big Data.