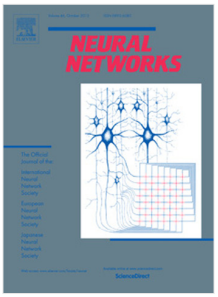
Accepted Manuscript



A systematic study of the class imbalance problem in convolutional neural networks

Mateusz Buda, Atsuto Maki, Maciej A. Mazurowski

PII: S0893-6080(18)30210-7
DOI: https://doi.org/10.1016/j.neunet.2018.07.011
Reference: NN 3994

To appear in: *Neural Networks*

Received date : 17 January 2018
Revised date : 26 May 2018
Accepted date : 20 July 2018

Please cite this article as: Buda, M., Maki, A., Mazurowski, M.A., A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* (2018), https://doi.org/10.1016/j.neunet.2018.07.011

*Title Page (With all author details listed)

A systematic study of the class imbalance problem in convolutional neural networks

Mateusz Buda[1, 2], Atsuto Maki[2], Maciej A. Mazurowski[1, 3]

{buda, atsuto}@kth.se, maciej.mazurowski@duke.edu

[1] Duke University Medical Center
Department of Radiology
2301 Erwin Road
Box 3808
Durham, NC 27710, USA

[2] School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
Teknikringen 14
11428 Stockholm, Sweden

[3] Department of Electrical and Computer Engineering
Edmund T. Pratt Jr. School of Engineering
Duke University
Box 90291
Durham, NC 27708, USA

Corresponding author:
Mateusz Buda
Carl E. Ravin Advanced Imaging Laboratories
Duke University
2424 Erwin Road, Hock Plaza, Suite 302
Durham, NC 27705, USA
Tel: 919-717-7178
E-mail: buda@kth.se

A systematic study of the class imbalance problem in convolutional neural networks

Mateusz Buda[a,b,*], Atsuto Maki[b], Maciej A. Mazurowski[a,c]

[a]*Department of Radiology, Duke University School of Medicine, Durham, NC, USA*
[b]*School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden*
[c]*Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA*

Abstract

In this study, we systematically investigate the impact of class imbalance on classification performance of convolutional neural networks (CNNs) and compare frequently used methods to address the issue. Class imbalance is a common problem that has been comprehensively studied in classical machine learning, yet very limited systematic research is available in the context of deep learning. In our study, we use three benchmark datasets of increasing complexity, MNIST, CIFAR-10 and ImageNet, to investigate the effects of imbalance on classification and perform an extensive comparison of several methods to address the issue: oversampling, undersampling, two-phase training, and thresholding that compensates for prior class probabilities. Our main evaluation metric is area under the receiver operating characteristic curve (ROC AUC) adjusted to multi-class tasks since overall accuracy metric is associated with notable difficulties in the context of imbalanced data. Based on results from our experiments we conclude that (i) the effect of class imbalance on classification performance is detrimental; (ii) the method of addressing class imbalance that emerged as dominant in almost all analyzed scenarios was oversampling; (iii) oversampling should be applied to the level that completely eliminates the imbalance, whereas the optimal undersampling ratio depends on the extent of imbalance; (iv) as opposed to some classical machine learning models, oversampling does not cause overfitting of CNNs; (v) thresholding should be applied to compensate for prior class probabilities when overall number of properly classified cases is of interest.

Keywords: Class Imbalance, Convolutional Neural Networks, Deep Learning, Image Classification

*Corresponding author
Email addresses: buda@kth.se (Mateusz Buda), atsuto@kth.se (Atsuto Maki), maciej.mazurowski@duke.edu (Maciej A. Mazurowski)

1. Introduction

Convolutional neural networks (CNNs) are gaining significance in a number of machine learning application domains and are currently contributing to the state of the art in the field of computer vision, which includes tasks such as object detection, image classification, and segmentation. They are also widely used in natural language processing or speech recognition where they are replacing or improving classical machine learning models [1]. CNNs integrate automatic feature extraction and discriminative classifier in one model, which is the main difference between them and traditional machine learning techniques. This property allows CNNs to learn hierarchical representations [2]. The standard CNN is built with fully connected layers and a number of blocks consisting of convolutions, activation function layer and max pooling [3, 4, 5]. The complex nature of CNNs requires a significant computational power for training and evaluation of the networks, which is addressed with the help of modern graphical processing units (GPUs).

A common problem in real life applications of deep learning based classifiers is that some classes have a significantly higher number of examples in the training set than other classes. This difference is referred to as class imbalance. There are plenty of examples in domains like computer vision [6, 7, 8, 9, 10], medical diagnosis [11, 12], fraud detection [13] and others [14, 15, 16] where this issue is highly significant and the frequency of one class (e.g., cancer) can be 1000 times less than another class (e.g., healthy patient). It has been established that class imbalance can have significant detrimental effect on training traditional classifiers [17] including multi-layer perceptrons [18]. It affects both convergence during the training phase and generalization of a model on the test set. While the issue very likely also affects deep learning, no systematic study on the topic is available.

Methods of dealing with imbalance are well studied for classical machine learning models [19, 17, 20, 18]. The most straightforward and common approach is the use of sampling methods. Those methods operate on the data itself (rather than the model) to increase its balance. Widely used and proven to be robust is oversampling [21]. Another option is undersampling. Naïve version, called random majority undersampling, simply removes a random portion of examples from majority classes [17]. The issue of class imbalance can be also tackled on the level of the classifier. In such case, the learning algorithms are modified, e.g. by introducing different weights to misclassification of examples from different classes [22] or explicitly adjusting prior class probabilities [23].

Some previous studies showed results on cost sensitive learning of deep neural networks [24, 25, 26]. New kinds of loss function for neural networks training were also developed [27]. Recently, a new method for CNNs was introduced that trains the network in two-phases in which the network is trained on the balanced data first and then the output layers are fine-tuned [28]. While little systematic analysis of imbalance and methods to deal with it is available for deep learning, researchers employ some methods that might be addressing the problem likely based on intuition, some internal tests, and systematic results available for traditional machine learning. Based on our review of literature, the method most commonly applied in deep learning is oversampling.

The rest of this paper is organized as follows. Section 2 gives an overview of methods

to address the problem of imbalance. In Section 3 we describe the experimental setup. It provides details about compared methods, datasets and models used for evaluation. Then, in Section 4 we present the results from our experiments and compare methods. Finally, Section 5 concludes the paper.

2. Methods for addressing imbalance

Methods for addressing class imbalance can be divided into two main categories [29]. The first category is data level methods that operate on training set and change its class distribution. They aim to alter dataset in order to make standard training algorithms work. The other category covers classifier (algorithmic) level methods. These methods keep the training dataset unchanged and adjust training or inference algorithms. Moreover, methods that combine the two categories are available. In this section we give an overview of commonly used approaches in both classical machine learning models and deep neural networks.

2.1. Data level methods

Oversampling. One of the most commonly used method in deep learning [16, 30, 31, 32]. The basic version of it is called random minority oversampling, which simply replicates randomly selected samples from minority classes. It has been shown that oversampling is effective, yet it can lead to overfitting [33, 34]. A more advanced sampling method that aims to overcome this issue is SMOTE [33]. It augments artificial examples created by interpolating neighboring data points. Some extensions of this technique were proposed, for example focusing only on examples near the boundary between classes [35]. Another type of oversampling approach uses data preprocessing to perform more informed oversampling. Cluster-based oversampling first clusters the dataset and then oversamples each cluster separately [36]. This way it reduces both between-class and within-class imbalance. DataBoost-IM, on the other hand, identifies difficult examples with boosting preprocessing and uses them to generate synthetic data [37]. An oversampling approach specific to neural networks optimized with stochastic gradient descent is class-aware sampling [38]. The main idea is to ensure uniform class distribution of each mini-batch and control the selection of examples from each class.

Undersampling. Another popular method [16] that results in having the same number of examples in each class. However, as opposed to oversampling, examples are removed randomly from majority classes until all classes have the same number of examples. While it might not appear intuitive, there is some evidence that in some situations undersampling can be preferable to oversampling [39]. A significant disadvantage of this method is that it discards a portion of available data. To overcome this shortcoming, some modifications were introduced that more carefully select examples to be removed. E.g. one-sided selection identifies redundant examples close to the boundary between classes [40]. A more general approach than undersampling is data decontamination that can involve relabeling of some examples [41, 42].

2.2. Classifier level methods

Thresholding. Also known as threshold moving or post scaling, adjusts the decision threshold of a classifier. It is applied in the test phase and involves changing the output class probabilities. There are many ways in which the network outputs can be adjusted. In general, the threshold can be set to minimize arbitrary criterion using an optimization algorithm [23]. However, the most basic version simply compensates for prior class probabilities [43]. These are estimated for each class by its frequency in the imbalanced dataset before sampling is applied. It was shown that neural networks estimate Bayesian a posteriori probabilities [43]. That is, for a given datapoint x, their output for class i implicitly corresponds to

$$y_i(x) = p(i|x) = \frac{p(i) \cdot p(x|i)}{p(x)}.$$

Therefore, correct class probabilities can be obtained by dividing the network output for each class by its estimated prior probability $p(i) = \frac{|i|}{\sum_k |k|}$, where $|i|$ denotes the number of unique examples in class i.

Cost sensitive learning. This method assigns different cost to misclassification of examples from different classes [44]. With respect to neural networks it can be implemented in various ways. One approach is threshold moving [22] or post scaling [23] that are applied in the inference phase after the classifier is already trained. Similar strategy is to adapt the output of the network and also use it in the backward pass of backpropagation algorithm [45]. Another adaptation of neural network to be cost sensitive is to modify the learning rate such that higher cost examples contribute more to the update of weights. And finally we can train the network by minimizing the misclassification cost instead of standard loss function [45]. The results of this approach are equivalent to oversampling [22, 26] described above and therefore this method will not be implemented in our study.

One-class classification. In the context of neural networks it is usually called novelty detection. This is a concept learning technique that recognizes positive instances rather than discriminating between two classes. Autoencoders used for this purpose are trained to perform autoassociative mapping, i.e. identity function. Then, the classification of a new example is made based on a reconstruction error between the input and output patterns, e.g. absolute error, squared sum of errors, Euclidean or Mahalanobis distance [46, 47, 48]. This method has proved to work well for extremely high imbalance when classification problem turns into anomaly detection [49].

Hybrid of methods. This is an approach that combines multiple techniques from one or both abovementioned categories. Widely used example is ensembling. It can be viewed as a wrapper to other methods. *EasyEnsemble* and *BalanceCascade* are methods that train a committee of classifiers on undersampled subsets [50]. SMOTEBoost, on the other hand, is a combination of boosting and SMOTE oversampling [51]. Recently introduced and successfully applied to CNN training for brain tumor segmentation, is two-phase training [28]. Even though the task was image segmentation, it was approached as a pixel level classification. The method involves network pre-training on balanced dataset and then fine-tuning the last output layer before softmax on the original, imbalanced data.

3. Experiments

3.1. Forms of imbalance

Class imbalance can take many forms particularly in the context of multiclass classification, which is typical in CNNs. In some problems only one class might be underrepresented or overrepresented and in other every class will have a different number of examples. In this study we define and investigate two types of imbalance that we believe are representative of most of the real-world cases.

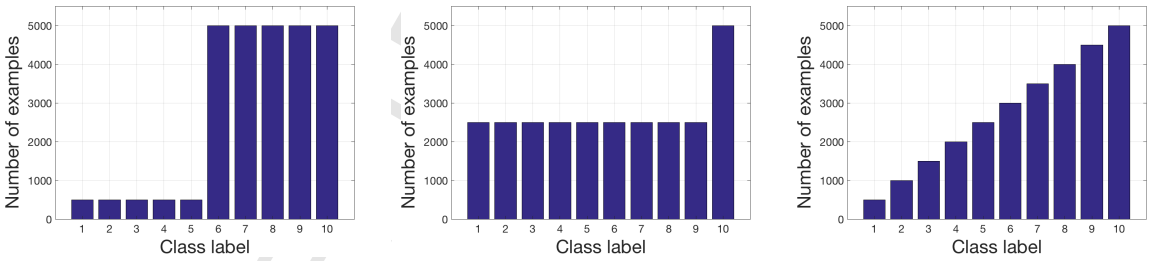
The first type is *step imbalance*. In *step imbalance*, the number of examples is equal within minority classes and equal within majority classes but differs between the majority and minority classes. This type of imbalance is characterized by two parameters. One is the fraction of minority classes defined by

$$\mu = \frac{|\{i \in \{1, \ldots, N\} : C_i \text{ is minority}\}|}{N}, \tag{1}$$

where C_i is a set of examples in class i and N is the total number of classes. The other parameter is a ratio between the number of examples in majority classes and the number of examples in minority classes defined as follows.

$$\rho = \frac{\max_i\{|C_i|\}}{\min_i\{|C_i|\}} \tag{2}$$

An example of this type of imbalance is the situation when among the total of 10 classes, 5 of them have 500 training examples and another 5 have 5 000. In this case $\rho = 10$ and $\mu = 0.5$, as shown in Figure 1a. A dataset with the same number of examples in total that has smaller imbalance ratio, corresponding to parameter $\rho = 2$, but more classes being minority, $\mu = 0.9$, is presented in Figure 1b.



(a) $\rho = 10, \mu = 0.5$ (b) $\rho = 2, \mu = 0.9$ (c) $\rho = 10$

Figure 1: Example distributions of imbalanced set together with corresponding values of parameters ρ and μ for *step imbalance* (a - b) and ρ for *linear imbalance* (c).

The second type of imbalance we call *linear imbalance*. We define it with one parameter that is a ratio between the maximum and minimum number of examples among all classes, as in Equation 2 for imbalance ratio in *step imbalance*. However, the number of examples in the remaining classes is interpolated linearly such that the difference between consecutive pairs of classes is constant. An example of linear imbalance distribution with $\rho = 10$ is shown in Figure 1c.

3.2. Methods of addressing imbalance compared in this study

In total, we examine seven methods to handle CNN training on a dataset with class imbalance which cover most of the commonly used approaches in the context of deep learning:

1. Random minority oversampling
2. Random majority undersampling
3. Two-phase training with pre-training on randomly oversampled dataset
4. Two-phase training with pre-training on randomly undersampled dataset
5. Thresholding with prior class probabilities
6. Oversampling with thresholding
7. Undersampling with thresholding

We examine two variants of two-phase training method. One on oversampled and the other on undersampled dataset. For the second phase, we keep the same hyperparameters and learning rate decay policy as in the first phase. Only the base learning rate from the first phase is multiplied by the factor of 10^{-1}. Regarding thresholding, this method originally uses the imbalanced training set to train a neural network. We, in addition, combine it with oversampling and undersampling.

Selected methods are representative of the available approaches. Sampling can be used to explicitly incorporate cost of the examples by their appearance. It makes them one of many implementations of cost-sensitive learning [22]. Thresholding is another way of applying cost-sensitiveness by moving the output threshold such that higher cost examples are harder to misclassify. Ensemble methods require training of multiple classifiers. Because of considerable time needed to train deep models, it is often not practical and may be even infeasible to train multiple deep neural networks. One-class methods have a very limited application to datasets with extremely high imbalance. Moreover, they are applied to anomaly detection problem that is beyond the scope of our study.

Importantly, we focused on methods that are widely used and relatively straightforward to implement as our aim is to draw conclusions that will be practical and serve as a guidance to a large number of deep learning researchers and engineers.

3.3. Datasets and models

In our study, we used three benchmark datasets: MNIST [52], CIFAR-10 [53] and ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 [54]. All of them are provided with a split on training and test set that are both labelled. For each dataset we choose different model with a set of hyperparameters used for its training that is known to perform well based on the literature. Datasets together with their corresponding models are of increasing complexity. This allows us to draw some conclusions on simple task and then verify how they scale to more complex ones.

All networks for the same dataset were trained with equal number of iterations. It means that the number of epochs differs between the imbalanced versions of dataset. This way we keep the number of weights' updates constant. Also, all networks were trained from a random initialization of weights and no pretraining was applied. An overview of some information about the datasets and their corresponding models is given in Table 1. All experiments were implemented in the deep learning framework *Caffe* [55].

Dataset	Image dimensions			No. classes	Images per class		CNN model
	Width	Height	Depth		Training	Test	
MNIST	28	28	1	10	5 000	1 000	LeNet-5
CIFAR-10	32	32	3	10	5 000	1 000	All-CNN
ILSVRC-2012	≥ 256	≥ 256	3	1 000	1 000	50	ResNet-10

Table 1: Summary of the used datasets. The number of images per class refers to the perfectly balanced subsets used for experiments. Provided image dimensions for ImageNet are given after rescaling.

3.3.1. MNIST

MNIST is considered simple and solved problem that involves digits images classification. The dataset consists of grayscale images of size 28×28. There are ten classes corresponding to digits from 0 to 9. The number of examples per class in the original training dataset ranges from 5421 in class 5 to 6742 in class 1. In artificially imbalanced versions we uniformly at random subsample each class to contain no more than 5 000 examples.

Layer	Data dimensions			Kernel size	Stride
	Width	Height	Depth		
Input	28	28	1	-	-
Convolution	24	24	20	5	1
Max Pooling	12	12	20	2	2
Convolution	8	8	50	5	1
Max Pooling	4	4	50	2	2
Fully Connected	1	1	500	-	-
ReLU	1	1	500	-	-
Fully Connected	1	1	10	-	-
Softmax	1	1	10	-	-

Table 2: Architecture of LeNet-5 CNN used in MNIST experiments.

The CNN model that we use for MNIST is the modern version of LeNet-5 [52]. The network architecture is presented in Table 2. All networks for this dataset were trained for 10 000 iterations. Optimization algorithm is stochastic gradient descent (SGD) with momentum value of $\mu = 0.9$ [56]. The learning rate decay policy is defined as $\eta_t = \eta_0 \cdot (1 + \gamma \cdot t)^{-\alpha}$, where $\eta_0 = 0.01$ is a base learning rate, $\gamma = 0.0001$ and $\alpha = 0.75$ are decay parameters and t is the current iteration. Furthermore, we used a batch size of 64 and a weight decay value of $\lambda = 0.0005$. Network weights were initialized randomly with uniform distribution and Xavier variance [57] whereas the biases were initialized with zero. No data augmentation was used. Test error of the model trained as described above on the original MNIST dataset was below 1%.

Experiments on MNIST dataset are performed on the following imbalance parameters

space. For *linear imbalance* we test values of $\rho \in \{10, 25, 50, 100, 250, 500, 1\,000, 2\,500, 5\,000\}$. For *step imbalance* the set of ρ values is the same and for each we use all possible number of minority classes from 1 to 9, which corresponds to $\mu \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The experiment for each combination of parameters is repeated 50 times. Every time the subset of minority classes is randomized. This way, we have created 4 050 artificially imbalanced training sets for *step imbalance* and 450 for linear imbalance. As we have evaluated four methods that require training a model, the total number of trained networks, including baseline, is 22 500.

3.3.2. CIFAR-10

CIFAR-10 is a significantly more complex image classification problem than MNIST. It contains 32×32 color images with ten classes of natural objects. It does not have any natural imbalance at all. There are exactly 5 000 training and 1 000 test examples in each class. We do not use any data augmentation but follow standard preprocessing comprising global contrast normalization and ZCA whitening [58].

	Data dimensions					
Layer	Width	Height	Depth	Kernel size	Stride	Padding
Input	32	32	3	-	-	-
Dropout (0.2)	32	32	3	-	-	-
2×(Convolution + ReLU)	32	32	96	3	1	1
Convolution + ReLU	16	16	96	3	2	1
Dropout (0.5)	16	16	96	-	-	-
2×(Convolution + ReLU)	16	16	192	3	1	1
Convolution + ReLU	8	8	192	3	2	1
Dropout (0.5)	8	8	192	-	-	-
Convolution + ReLU	6	6	192	3	1	0
Convolution + ReLU	6	6	192	1	1	0
Convolution + ReLU	6	6	10	1	1	0
Average Pooling	1	1	10	6	-	-
Softmax	1	1	10	-	-	-

Table 3: Architecture of All-CNN used in CIFAR-10 experiments.

For CIFAR-10 experiments we use one of the best performing type of CNN model on this dataset, i.e. All-CNN [59]. The network architecture is presented in Table 3. The networks were trained for 70 000 iterations using SGD with momentum $\mu = 0.9$. The base learning rate was multiplied by a fixed multiplier of 0.1 after 40 000, 50 000 and 60 000 iterations. The number of examples in a batch was 256 and a weight decay value was $\lambda = 0.001$. Network weights were initialized with Xavier procedure and the biases set to zero. Test error of the model trained as described above on the original CIFAR-10 dataset was 9.75%.

We have found the network training to be quite sensitive to initialization and the choice of base learning rate. Sometimes the network gets stuck in a very poor local minimum.

Also, for more imbalanced datasets the training required lower base learning rate to train at all. Therefore, for each case we were searching for the best one from the fixed set $\eta_0 \in \{0.05, 0.005, 0.0005, 0.00005\}$. Similar procedure was used by the authors of the model architecture [59]. Moreover, each training was repeated twice on the same dataset. For a particular method and imbalanced dataset, we pick the model with the best score on the test set over all eight runs.

The network architecture does not have any fully connected layers. Therefore, during the fine-tuning in two-phase training method we update the weights of two last convolutional layers with kernels of size 1.

The imbalance parameters space used in CIFAR-10 experiments is considerably sparser than the one used for MNIST due to the significantly longer time required to train one network. The set of tested values was narrowed to make the experiment run in a reasonable time. For *linear* and *step imbalance*, we test values of $\rho \in \{2, 10, 20, 50\}$. In *step imbalance*, for each value of ρ, the set of values of parameter μ was $\mu \in \{0.2, 0.5, 0.8\}$, which corresponds to having two, five and eight minority classes respectively. And for all the cases, the classes chosen to be minority were the ones with the lowest label value. It means that for a fixed number of minority classes the same classes were always picked as minority. Also, all of them were included in a larger set of minority classes. In total we trained 640 networks on this dataset.

3.3.3. ImageNet

For evaluation we use a ILSVRC-2012 competition subset of ImageNet, widely used as a benchmark to compare classifiers' performance. The number of examples in majority classes was reduced from 1 200 to 1 000. Classes with less than 1 000 cases were always chosen as a minority ones for imbalanced subsets. The only data preprocessing applied is resizing such that the smaller dimension is 256 pixels long and the aspect ratio is preserved. During training, as input we use a randomly cropped 224×224 pixel square patch and a single centered crop in a test phase. Moreover, during training we randomly mirror images, but there is no color, scale or aspect ratio augmentation.

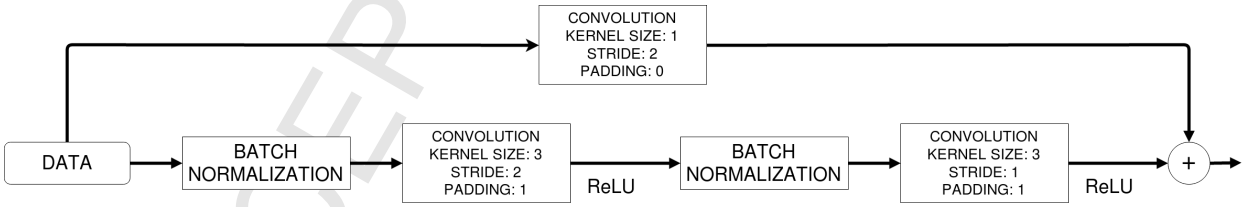


Figure 2: Architecture of a single residual block in ResNet used in ILSVRC-2012 experiments.

A model architecture employed for this dataset is ResNet-10 [60], i.e. a residual network [61] with batch normalization layers that are known to accelerate deep networks training [62]. It consists of four residual blocks that give us nine convolutional layers and one fully connected. The first residual block outputs data tensor of depth 64 and then each one increases it by a factor of two. Fully connected layer outputs 1 000 values to softmax that

transforms them to class probabilities. The architecture of one residual block is presented in Figure 2.

The networks were trained for 320 000 iterations using SGD with momentum $\mu = 0.9$. The base learning rate is set to $\eta_0 = 0.1$ and decays linearly to 0 in the last iteration. The number of examples in a batch was 256 and a weight decay $\lambda = 0.0001$. Network weights were initialized with Kaiming (also known as MSRA) initialization procedure [63]. Top-1 test error of the model trained as described above on the original ILSVRC-2012 dataset was 62.56% and 99.50 multi-class ROC AUC for a single centered crop.

We have chosen relatively small ResNet for the sake of faster training but without loss of generality[1]. We test only one case of small and two cases of large *step imbalance* and run it on the baseline, undersampling and oversampling methods. Specifically, all three *step imbalanced* subsets are defined with $\mu = 0.1, \rho = 10$, $\mu = 0.8, \rho = 50$ and $\mu = 0.9, \rho = 100$. They correspond to 100 minority classes with imbalance ratio of 10, 800 minority classes with imbalance of 50, and 900 minority classes with imbalance ratio of 100 respectively. Moreover, for the highest imbalance, we train three networks for each method with randomized selection of minority classes and subsampled set of examples in each class. This is done in order to estimate variability in performance of methods. In total, this gives us 15 ResNet-10 networks trained on five artificially imbalanced subsets of ILSVRC-2012.

3.4. Evaluation metrics and testing

The metric that is most widely used to evaluate a classifier performance in the context of multiclass classification with CNNs is overall accuracy which is the proportion of test examples that were correctly classified. However, it has some significant and long acknowledged limitations, particularly in the context of imbalanced datasets [19]. Specifically, when the test set is imbalanced, accuracy will favor classes that are overrepresented in some cases leading to highly misleading assessment. An example of this is a situation when the majority class represents 99% of all cases and the classifier assigns the label of the majority class to all test cases. A misleading accuracy of 99% will be assigned to a classifier that has a very limited use. Another issue might arise when the test set is balanced and a training set is imbalanced. This might result in a situation when a decision threshold is moved to reflect the estimated class prior probabilities and cause a low accuracy measure in the test set while the true discriminative power of the classifier does not change.

A measure that addresses these issues is area under the receiver operating characteristic curve (ROC AUC) [64] which is a plot of the false positive rate to the true positive rate for all possible prediction thresholds. We used a specific implementation of the ROC AUC available in scikit-learn python package [65]. It calculates sensitivities and specificities at all thresholds defined by the responses of the classifier in the test set followed by the AUC calculation using the trapezoid rule. ROC AUC is a well-studied and sound measure of discrimination [66] and has been widely used as an evaluation metric for classifiers. ROC has also been used to compare performance of classifiers trained on imbalanced datasets [18, 20]. Since the basic version of ROC is only suitable for binary classification, we use a multi-class

[1]It takes five days to train one ResNet-10 network on Nvidia GTX 1070 GPU.

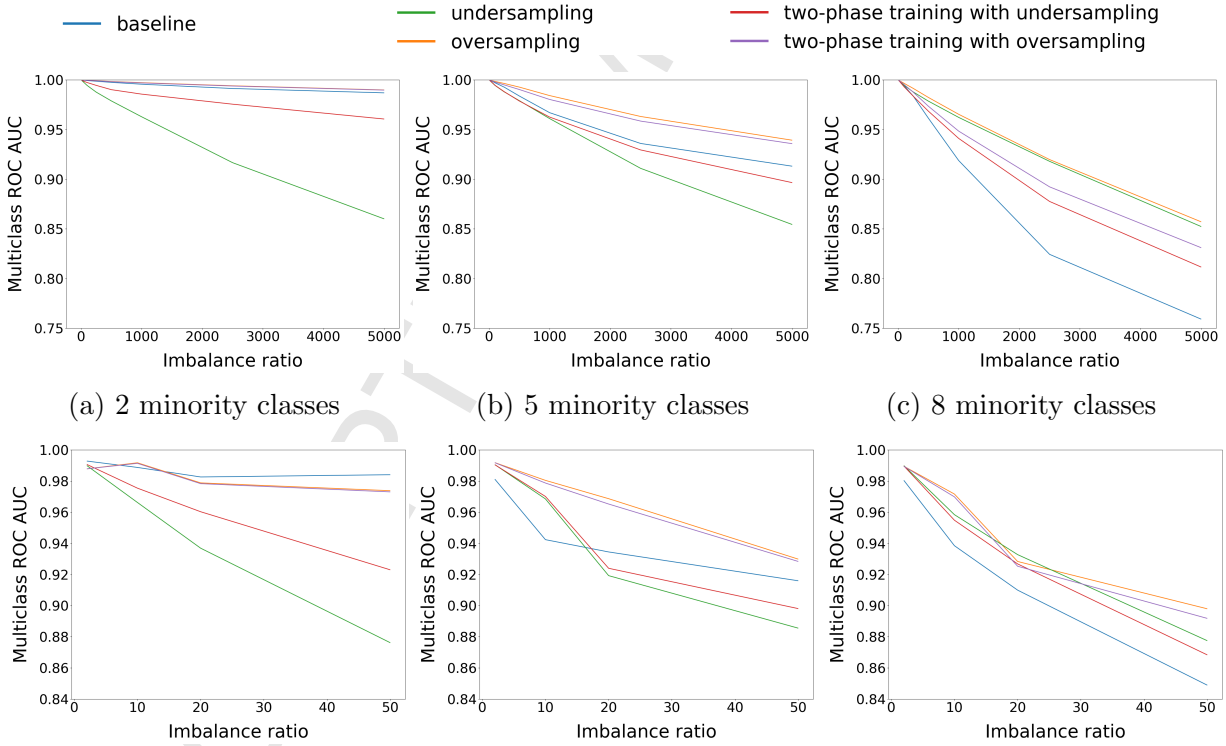
modification of it [67]. The multi-class ROC is calculated by taking the average of AUCs obtained independently for each class for the binary classification task of distinguishing a given class from all the other classes.

Test set of all used datasets has equal number of examples in each class. Usually, it is assumed that the class distribution of a test set follows the one of a training set. We do not change a test set to match artificially imbalanced training set. The reason is that the score achieved by each classifier on the same test set is more comparable and the largest number of cases in each of the classes provides the most accurate performance estimation.

4. Results

4.1. Effects of class imbalance on classification performance and comparison of methods to address imbalance

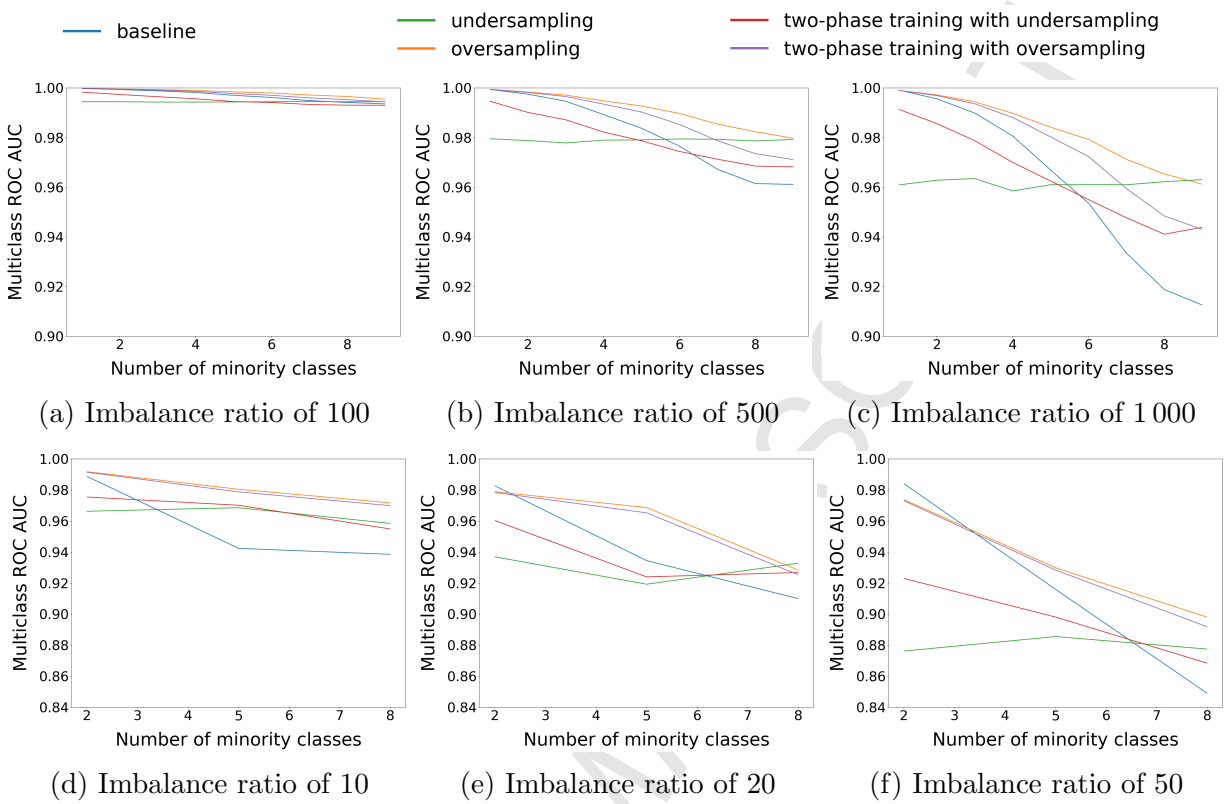
The results showing the impact of class imbalance on classification performance and comparison of methods for addressing imbalance are shown in Figures 3 and 4. Figure 3 shows the results with respect to multi-class ROC AUC for a fixed number of minority classes on MNIST and CIFAR-10. Figure 4 presents the result from the perspective of fixed ratio of imbalance, i.e. parameter ρ, for the same two datasets.



(a) 2 minority classes (b) 5 minority classes (c) 8 minority classes

(d) 2 minority classes (e) 5 minority classes (f) 8 minority classes

Figure 3: Comparison of methods with respect to multi-class ROC AUC on MNIST (a - c) and CIFAR-10 (d - f) for *step imbalance* with fixed number of minority classes.



(a) Imbalance ratio of 100 (b) Imbalance ratio of 500 (c) Imbalance ratio of 1 000

(d) Imbalance ratio of 10 (e) Imbalance ratio of 20 (f) Imbalance ratio of 50

Figure 4: Comparison of methods with respect to multi-class ROC AUC on MNIST (a - c) and CIFAR-10 (d - f) for *step imbalance* with fixed imbalance ratio.

Regarding the effect of class imbalance on classification performance, we observed the following. First, the deterioration of performance due to class imbalance is substantial. As expected, the increasing ratio of examples between majority and minority classes as well as the number of minority classes had a negative effect on performance of the resulting classifiers. Furthermore, by comparing the results from MNIST and CIFAR-10 we observed that the effect of imbalance is significantly stronger for the task with higher complexity. A similar drop in performance for MNIST and CIFAR-10 corresponded to approximately 100 times stronger level of imbalance in the MNIST dataset.

Regarding performance of different methods for addressing imbalance, in almost all of the situations oversampling emerged as the best method. It also showed notable improvement of performance over the baseline (i.e. do-nothing strategy) in majority of the situations and never showed a considerable decrease in performance for the two datasets analyzed in this section making it a clear recommendation for tasks similar to MNIST and CIFAR-10.

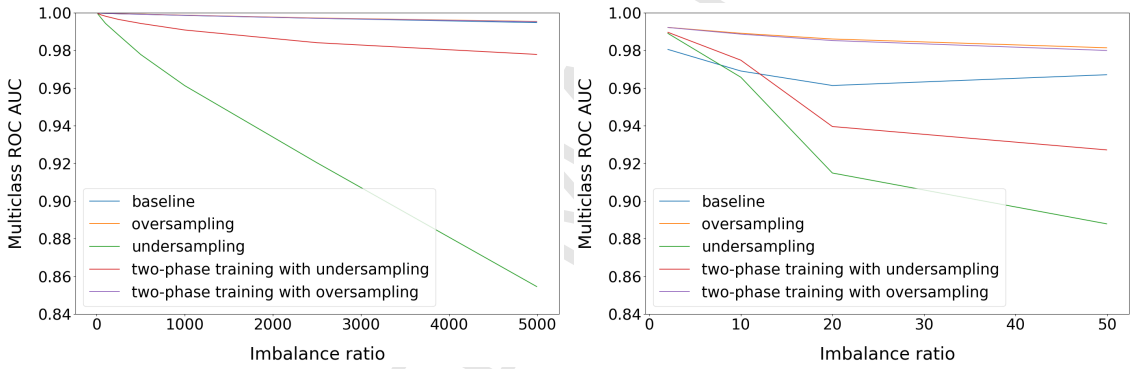
Undersampling showed a generally poor performance. In a large number of analyzed scenarios undersampling showed decrease in performance as compared to the baseline. In scenarios with a large proportion of minority classes undersampling showed some improvement over the baseline but never a notable advantage over oversampling (Figure 3).

For a fixed imbalance ratio undersampling is always trained on the subset of equal size.

As a result, its performance does not change with the number of minority classes. For both datasets and each case of imbalance ratio, the gap between undersampling and oversampling is the biggest for smaller number of minority classes and decreases with the number of minority classes, as shown in Figure 4. This is expected since with all classes being minority these two methods become equivalent.

Two-phase training methods with both undersampling and oversampling tend to perform between the baseline and their corresponding method (undersampling or oversampling). If the baseline is better than one of these methods, fine-tuning improves the original method. Otherwise, performance deteriorates. However, if the baseline is better, there is still no gain from using two-phase training method. As oversampling is almost always better than the baseline, fine-tuning always gives lower score.

The variability of patterns, visual structures, and objects in CIFAR-10 is considerably higher than in MNIST. For this reason, we run the *step imbalance* experiment three times on a reshuffled stratified training and test split to validate our results. Additional results are available in Appendix A.



(a) MNIST (b) CIFAR-10

Figure 5: Comparison of methods with respect to multi-class ROC AUC for *linear imbalance*.

In Figure 5 we show the results for *linear imbalance* on MNIST and CIFAR-10 datasets. The highest possible *linear imbalance* ratio for MNIST dataset is 5 000, which means only one example in the most underrepresented class. However, even in this case the decrease in performance according to multi-class ROC AUC score for the baseline model is not significant, as shown in Figure 5a. Nevertheless, oversampling improves the score on both datasets and for all tested values of ρ, whereas the score for undersampling decreases approximately linearly with imbalance ratio.

4.2. Results on ImageNet dataset

The results from experiments performed on ImageNet (ILSVRC-2012) dataset confirm the impact of imbalance on classifiers performance. Table 4 compares methods with respect to multi-class ROC AUC. The drop in performance for the largest tested imbalance was from 99 to 90, in terms of multi-class ROC AUC. The results confirm that the oversampling approach performs consistently better than undersampling approach across all scenarios.

A small decrease in performance as compared to baseline was observed for oversampling for extreme imbalances. Please note, however, that these results should be treated with caution and not as strong evidence that oversampling is inferior for highly complex tasks with extreme imbalance. The absolute difference in performance between three runs with respect to multi-class ROC AUC was even higher than 4 (for undersampling). Therefore, differences of 1 - 2 might be due to variability of results between different runs of neural networks. Moreover, the highest tested imbalanced training set was only about 10% of the original ILSVRC-2012 introducing confounding issues such as the optimal training hyperparameters for this significantly changed dataset. Therefore, while these results indicate that caution should be taken when any sampling technique is applied to highly complex tasks with extreme imbalances, it needs a more extensive study devoted to this specific issue.

Method	$\mu = 0.1, \rho = 10$	$\mu = 0.8, \rho = 50$	$\mu = 0.9, \rho = 100$		
Baseline	99.41	96.31	90.74	90.46	90.05
Oversampling	99.35	95.06	88.38	88.39	88.17
Undersampling	96.85	94.98	88.35	84.08	83.74

Table 4: Comparison of results on ImageNet with respect to multi-class ROC AUC.

4.3. Separation of effects from reduced number of examples and class imbalance

An important question that needs to be considered in the context of our study is whether the decrease in performance for imbalanced datasets is merely caused by the fact that our imbalanced datasets simply had fewer training examples or is it truly caused by the fact that the datasets are imbalanced.

First, we notice that oversampling method uses the same amount of data as the baseline. It only eliminates the imbalance which is enough to improve the performance in almost all the cases. Still, it does not reach the performance of a classifier trained on the original dataset. This is an indication that the effect of imbalance is not trivial.

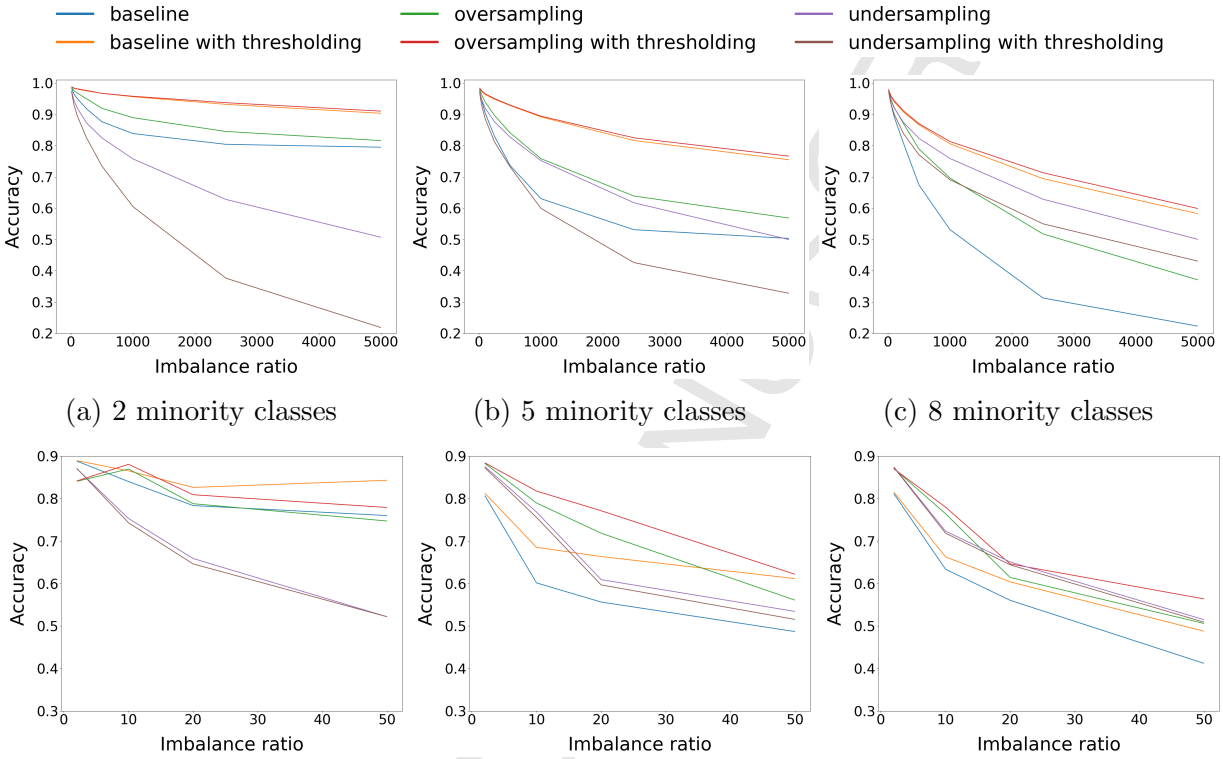
Second, for some cases undersampling, which reduces the total number of cases performs better than the baseline (see Figures 3c and 3f). Moreover, there are even cases when undersampling can perform on a par with oversampling. It means that, between two sampling methods that eliminate imbalance, even using fewer data can be comparable.

In addition, for the same value of parameter ρ we have equal number of examples in the training set for *linear imbalance* and *step imbalance* with $\mu = 0.5$, which corresponds to half of the classes being minority. The drop in performance is much higher for *step imbalance*. This additionally demonstrates that not only the total number of examples matters but also its distribution between classes.

4.4. Improving accuracy score with multi-class thresholding

While our focus is on ROC AUC, we also provide the evaluation of the methods based on overall accuracy measure with results on step imbalance shown in Figure 6. As explained

in Section 3.4, accuracy has some known limitations and in some scenarios does not reflect the discriminative power of a classifier but rather the prevalence of classes in the training or test set. Nevertheless, it is still commonly used evaluation score [16] and therefore we provide some results according to this metric.



(a) 2 minority classes (b) 5 minority classes (c) 8 minority classes

(d) 2 minority classes (e) 5 minority classes (f) 8 minority classes

Figure 6: Comparison of methods with respect to accuracy on MNIST (a - c) and CIFAR-10 (d - f) for *step imbalance* with fixed number of minority classes.

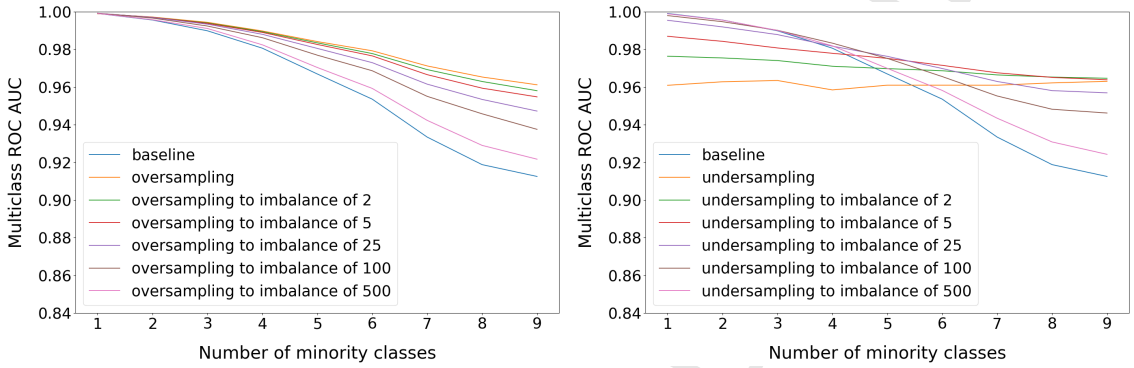
Our results show that thresholding is an appropriate approach to take to offset the prior probabilities of different classes learned by a network based on imbalanced datasets and provided an improvement in overall accuracy. In general, thresholding worked particularly well when applied jointly with oversampling.

Please note that thresholding does not have an actual effect on the ability of the classifier to discriminate between a given class from another but rather helps to find a threshold on the network output that guarantees a large number of correctly classified cases. In terms of ROC, multiplying a decision variable by any positive number does not change the area under the ROC curve. However, finding an optimal operating point on the ROC curve is important when the overall number of correctly classified cases is of interest.

4.5. Undersampling and oversampling to smaller imbalance ratio

The default version of oversampling is to increase the number of cases in the minority classes so that the number matches the majority classes. Similarly, the default of under-

sampling is to decrease the number of cases in the majority classes to match the minority classes. However, a more moderate version of these algorithms could be applied. For the case of MNIST with imbalance ratio of 1 000 we have tried to gradually decrease the imbalance with oversampling and undersampling. The results are shown in Figure 7.



(a) Oversampling (b) Undersampling

Figure 7: Comparison of oversampling and undersampling to reduced imbalance ratios on MNIST with original imbalance of 1 000.

The results show that the default version of oversampling was always the best. Any reduction of imbalance improves the score regardless of the number of minority classes, as shown in Figure 7a. For undersampling, in some cases of moderate number of minority classes, intermediate levels of undersampling performed better than both full undersampling and the baseline.

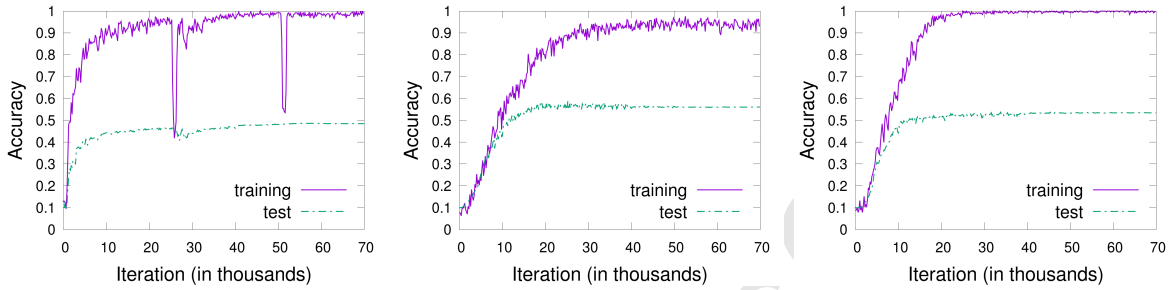
Moreover, comparing undersampling and oversampling to reduced level of imbalance, we can notice that for each case of oversampling there is a level to which we can apply undersampling and achieve equivalent performance. However, that level is not known a priori rendering oversampling still the method of choice.

4.6. Generalization of sampling methods

In some cases undersampling and oversampling perform similarly. In those cases, one would probably prefer the model that generalizes better. For classical machine learning models it was shown that oversampling can cause overfitting, especially for minority classes [33]. As we repeat small number of examples multiple times, the trained model fits them too well. Thus, according to this prior knowledge undersampling would be a better choice. The results from our experiments do not confirm this conclusion for convolutional neural networks.

In Figure 8 we compare the convergence of baseline and sampling methods for CIFAR-10 experiments with respect to accuracy. Both oversampling and undersampling methods helped to train a better classifier in terms of performance and generalization. They also made training more stable. As opposed to traditional machine learning methods, in this case oversampling did not lead to overfitting. The gap between accuracy on the training and test set does not increase with iterations for oversampling, Figure 8b. Furthermore, we validated this phenomenon in multiple additional scenarios for all analyzed datasets and

have not observed overfitting in any of these scenarios. The additional plots are included in Appendix B.



(a) Baseline (b) Oversampling (c) Undersampling

Figure 8: Comparison of networks convergence between baseline, oversampling and undersampling. Training on CIFAR-10 *step imbalanced* with 5 minority classes and imbalance ratio of 50.

5. Conclusions

In this study, we examined the impact of class imbalance on classification performance of convolutional neural networks and investigated the effectiveness of different methods of addressing the issue. We defined and parametrized two representative types of imbalance, i.e. step and linear. Then we subsampled MNIST, CIFAR-10 and ImageNet (ILSVRC-2012) datasets to make them artificially imbalanced. We have compared common sampling methods, basic thresholding, and two-phase training.

The observations from our experiments related to the class imbalance are as follows.

- The effect of class imbalance on classification performance is detrimental.
- The influence of imbalance on classification performance increases with the scale of a task.
- The impact of imbalance cannot be explained simply by the lower total number of training cases and depends on the distribution of examples among classes.

Regarding the choice of a method to handle CNN training on imbalanced dataset we conclude the following.

- The method that in most of the cases outperforms all others with respect to multi-class ROC AUC was oversampling.
- For extreme ratio of imbalance and large portion of classes being minority, undersampling performs on a par with oversampling. If training time is an issue, undersampling is a better choice in such a scenario since it dramatically reduces the size of the training set.

- To achieve the best accuracy, one should apply thresholding to compensate for prior class probabilities. A combination of thresholding with baseline and oversampling is the most preferable, whereas it should not be combined with undersampling.
- Oversampling should be applied to the level that completely eliminates the imbalance, whereas the optimal undersampling ratio depends on the extent of imbalance. The higher a fraction of minority classes in the imbalanced training set, the more imbalance ratio should be reduced.
- Oversampling does not cause overfitting of convolutional neural networks, as opposed to some classical machine learning models.

References

[1] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, Recent advances in convolutional neural networks, arXiv preprint arXiv:1512.07108.

[2] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European conference on computer vision, Springer, 2014, pp. 818–833.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural computation 1 (4) (1989) 541–551.

[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[5] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[6] G. Van Horn, O. Mac Aodha, Y. Song, A. Shepard, H. Adam, P. Perona, S. Belongie, The inaturalist challenge 2017 dataset, arXiv preprint arXiv:1707.06642.

[7] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, A. Torralba, Sun database: Large-scale scene recognition from abbey to zoo, in: Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, IEEE, 2010, pp. 3485–3492.

[8] B. A. Johnson, R. Tateishi, N. T. Hoan, A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees, International journal of remote sensing 34 (20) (2013) 6969–6982.

[9] M. Kubat, R. C. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Machine learning 30 (2-3) (1998) 195–215.

[10] O. Beijbom, P. J. Edmunds, D. I. Kline, B. G. Mitchell, D. Kriegman, Automated annotation of coral reef survey images, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1170–1177.

[11] J. W. Grzymala-Busse, L. K. Goodwin, W. J. Grzymala-Busse, X. Zheng, An approach to imbalanced data sets based on changing rule strength, in: Rough-Neural Computing, Springer, 2004, pp. 543–553.

[12] B. Mac Namee, P. Cunningham, S. Byrne, O. I. Corrigan, The problem of bias in training data in regression problems in medical decision support, Artificial intelligence in medicine 24 (1) (2002) 51–70.

[13] P. K. Chan, S. J. Stolfo, Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection., in: KDD, Vol. 1998, 1998, pp. 164–168.

[14] P. Radivojac, N. V. Chawla, A. K. Dunker, Z. Obradovic, Classification and knowledge discovery in protein databases, Journal of Biomedical Informatics 37 (4) (2004) 224–239.

[15] C. Cardie, N. Howe, Improving minority class prediction using case-specific feature weights, in: ICML, 1997, pp. 57–65.

[16] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: Review of methods and applications, Expert Systems with Applications.

[17] N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intelligent data analysis 6 (5) (2002) 429–449.
[18] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, G. D. Tourassi, Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance, Neural networks 21 (2) (2008) 427–436.
[19] N. V. Chawla, Data mining for imbalanced datasets: An overview, in: Data mining and knowledge discovery handbook, Springer, 2005, pp. 853–867.
[20] M. A. Maloof, Learning when data sets are imbalanced and when costs are unequal and unknown, in: ICML-2003 workshop on learning from imbalanced data sets II, Vol. 2, 2003, pp. 2–1.
[21] C. X. Ling, C. Li, Data mining for direct marketing: Problems and solutions., in: KDD, Vol. 98, 1998, pp. 73–79.
[22] Z.-H. Zhou, X.-Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Transactions on Knowledge and Data Engineering 18 (1) (2006) 63–77.
[23] S. Lawrence, I. Burns, A. Back, A. C. Tsoi, C. L. Giles, Neural network classification and prior class probabilities, in: Neural networks: tricks of the trade, Springer, 1998, pp. 299–313.
[24] S. H. Khan, M. Bennamoun, F. Sohel, R. Togneri, Cost sensitive learning of deep feature representations from imbalanced data, arXiv preprint arXiv:1508.03422.
[25] V. Raj, S. Magg, S. Wermter, Towards effective classification of imbalanced data with convolutional neural networks, in: IAPR Workshop on Artificial Neural Networks in Pattern Recognition, Springer, 2016, pp. 150–162.
[26] Y.-A. Chung, H.-T. Lin, S.-W. Yang, Cost-aware pre-training for multiclass cost-sensitive deep learning, arXiv preprint arXiv:1511.09337.
[27] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, P. J. Kennedy, Training deep neural networks on imbalanced data sets, in: Neural Networks (IJCNN), 2016 International Joint Conference on, IEEE, 2016, pp. 4368–4374.
[28] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, H. Larochelle, Brain tumor segmentation with deep neural networks, Medical image analysis 35 (2017) 18–31.
[29] H. He, E. A. Garcia, Learning from imbalanced data, IEEE Transactions on knowledge and data engineering 21 (9) (2009) 1263–1284.
[30] G. Levi, T. Hassner, Age and gender classification using convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 34–42.
[31] A. Janowczyk, A. Madabhushi, Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases, Journal of pathology informatics 7.
[32] N. Jaccard, T. W. Rogers, E. J. Morton, L. D. Griffin, Detection of concealed cars in complex cargo x-ray imagery using deep learning, Journal of X-Ray Science and Technology (2016) 1–17.
[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: synthetic minority over-sampling technique, Journal of artificial intelligence research 16 (2002) 321–357.
[34] K.-J. Wang, B. Makond, K.-H. Chen, K.-M. Wang, A hybrid classifier combining smote with pso to estimate 5-year survivability of breast cancer patients, Applied Soft Computing 20 (2014) 15–24.
[35] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, Advances in intelligent computing (2005) 878–887.
[36] T. Jo, N. Japkowicz, Class imbalances versus small disjuncts, ACM Sigkdd Explorations Newsletter 6 (1) (2004) 40–49.
[37] H. Guo, H. L. Viktor, Learning from imbalanced data sets with boosting and data generation: the databoost-im approach, ACM Sigkdd Explorations Newsletter 6 (1) (2004) 30–39.
[38] L. Shen, Z. Lin, Q. Huang, Relay backpropagation for effective learning of deep convolutional neural networks, in: European Conference on Computer Vision, Springer, 2016, pp. 467–482.
[39] C. Drummond, R. C. Holte, et al., C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling, in: Workshop on learning from imbalanced datasets II, Vol. 11, 2003, pp. 1–8.
[40] M. Kubat, S. Matwin, et al., Addressing the curse of imbalanced training sets: one-sided selection, in:

ICML, Vol. 97, Nashville, USA, 1997, pp. 179–186.
[41] J. Koplowitz, T. A. Brown, On the relation of performance to editing in nearest neighbor rules, Pattern Recognition 13 (3) (1981) 251–255.
[42] R. Barandela, E. Rangel, J. S. Sánchez, F. J. Ferri, Restricted decontamination for the imbalanced training sample problem, in: Iberoamerican Congress on Pattern Recognition, Springer, 2003, pp. 424–431.
[43] M. D. Richard, R. P. Lippmann, Neural network classifiers estimate bayesian a posteriori probabilities, Neural computation 3 (4) (1991) 461–483.
[44] C. Elkan, The foundations of cost-sensitive learning, in: International joint conference on artificial intelligence, Vol. 17, Lawrence Erlbaum Associates Ltd, 2001, pp. 973–978.
[45] M. Kukar, I. Kononenko, et al., Cost-sensitive learning with neural networks., in: ECAI, 1998, pp. 445–449.
[46] N. Japkowicz, C. Myers, M. Gluck, et al., A novelty detection approach to classification, in: IJCAI, Vol. 1, 1995, pp. 518–523.
[47] N. Japkowicz, S. J. Hanson, M. A. Gluck, Nonlinear autoassociation is not equivalent to pca, Neural computation 12 (3) (2000) 531–545.
[48] H. Sohn, K. Worden, C. R. Farrar, Novelty detection using auto-associative neural network, in: Symposium on Identification of Mechanical Systems: international mechanical engineering congress and exposition, New York, NY, 2001, pp. 573–580.
[49] H.-j. Lee, S. Cho, The novelty detection approach for different degrees of class imbalance, in: Neural Information Processing, Springer, 2006, pp. 21–30.
[50] X.-Y. Liu, J. Wu, Z.-H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39 (2) (2009) 539–550.
[51] N. V. Chawla, A. Lazarevic, L. O. Hall, K. W. Bowyer, Smoteboost: Improving prediction of the minority class in boosting, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2003, pp. 107–119.
[52] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE.
[53] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images.
[54] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International Journal of Computer Vision 115 (3) (2015) 211–252.
[55] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM international conference on Multimedia, ACM, 2014, pp. 675–678.
[56] N. Qian, On the momentum term in gradient descent learning algorithms, Neural networks 12 (1) (1999) 145–151.
[57] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks., in: Aistats, Vol. 9, 2010, pp. 249–256.
[58] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, Y. Bengio, Maxout networks., ICML (3) 28 (2013) 1319–1327.
[59] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: The all convolutional net, arXiv preprint arXiv:1412.6806.
[60] M. Simon, E. Rodner, J. Denzler, Imagenet pre-trained models with batch normalization, arXiv preprint arXiv:1612.01452.
[61] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[62] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.
[63] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015,

pp. 1026–1034.

[64] A. P. Bradley, The use of the area under the roc curve in the evaluation of machine learning algorithms, Pattern recognition 30 (7) (1997) 1145–1159.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (Oct) (2011) 2825–2830.

[66] C. X. Ling, J. Huang, H. Zhang, Auc: a statistically consistent and more discriminating measure than accuracy, in: IJCAI, Vol. 3, 2003, pp. 519–524.

[67] F. Provost, P. Domingos, Tree induction for probability-based ranking, Machine learning 52 (3) (2003) 199–215.