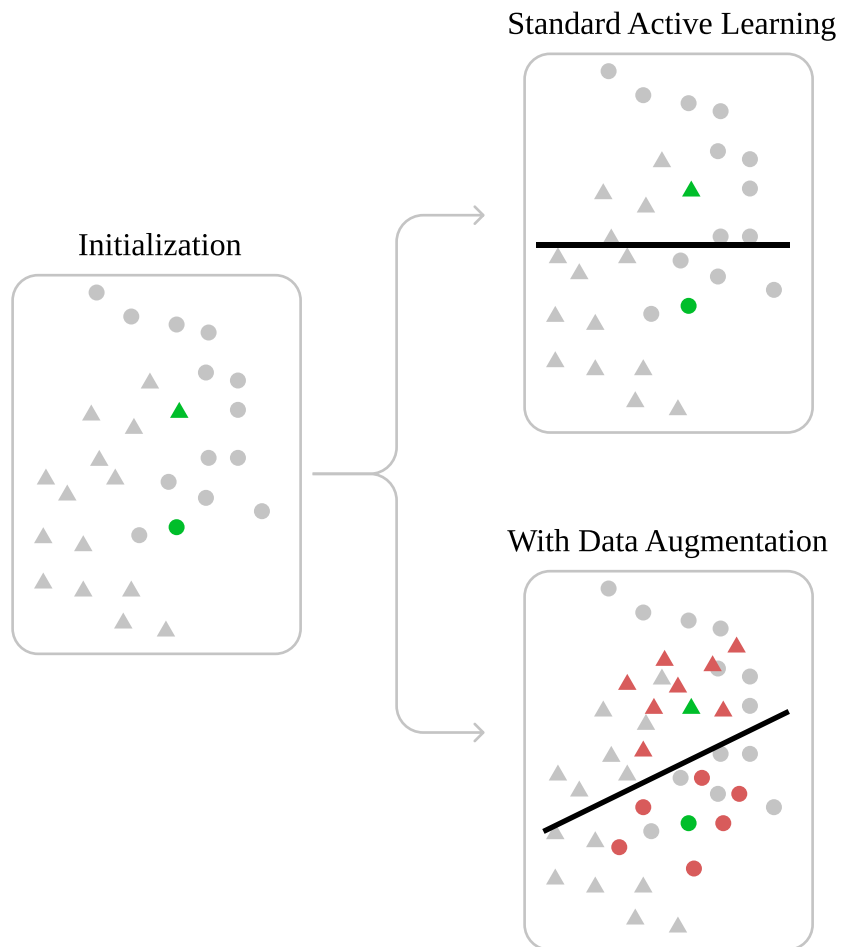


# Graphical Abstract

## Improving Active Learning Performance Through the Use of Data Augmentation

Joao Fonseca, Fernando Bacao



## Highlights

### **Improving Active Learning Performance Through the Use of Data Augmentation**

Joao Fonseca, Fernando Bacao

- We propose a new Active Learning framework that leverages hyperparameter optimization and data augmentation techniques;
- The use of data augmentation in Active Learning is sufficient to substantially improve the performance of an Active Learner, regardless of the choice of dataset/domain, classifier or metric.
- In most scenarios, the proposed method outperformed classifiers trained in fully supervised settings while using less data.

# Improving Active Learning Performance Through the Use of Data Augmentation

Joao Fonseca<sup>a</sup>, Fernando Bacao<sup>a</sup>

<sup>a</sup>*NOVA Information Management School, Universidade Nova de Lisboa, Campus de Campolide, Lisboa, 1070-312, Lisboa, Portugal*

---

## Abstract

Active Learning (AL) is a technique that is used to iteratively select unlabeled observations out of a large pool of unlabeled data to be labeled by a supervisor. Its focus is to find the unlabeled observations that, once labeled, will maximize the informativeness of the training dataset. However, the manual labeling of observations involves human resources with domain expertise, making it an expensive and time-consuming task. The literature describes various methods to improve the effectiveness of this process, but there is little research developed around the usage of artificial data sources in AL. In this paper we propose a new framework for AL, which allows for an effective use of artificial data. Our method implements a data augmentation policy that optimizes the generation of artificial instances to improve the AL process. We compare the proposed method to the standard framework as well as another active learning method that uses data augmentation. The models' performance was tested using 4 different classifiers, 2 AL-specific performance metrics and 3 classification performance metrics over 10 different datasets. We show that the proposed framework, using data augmentation, significantly improves the performance of AL, both in terms of classification performance and data selection efficiency.

*Keywords:* Active Learning, Data Augmentation, Oversampling

---

## 1. Introduction

The importance of training robust ML models with minimal data requirements is substantially increasing [1, 2, 3]. Although the growing amount of

valuable data sources and formats being developed and explored is affecting various domains [4], this data is often unlabeled. Only a small amount of the data being produced and stored can be useful for supervised learning tasks. Additionally, it’s important to note that labeling data for specific Machine Learning (ML) projects is often difficult and expensive, especially when data-intensive ML techniques are involved (*e.g.*, Deep Learning classifiers) [1]. In this scenario, labeling the full dataset becomes impractical, time-consuming, and expensive. Two different ML techniques attempt to address this problem: Semi-Supervised Learning (SSL) and Active Learning (AL). Even though they address the same problem, the two follow different approaches. SSL focuses on observations with the most certain predictions, whereas AL focuses on observations with the least certain predictions [5].

SSL attempts to use a small, predefined set of labeled and unlabeled data to produce a classifier with superior performance. This method uses the unlabeled observations to help define the classifier’s decision boundaries [6]. Simultaneously, the amount of labeled data required to reach a given performance threshold is also reduced. It is a special case of ML because it falls between the supervised and unsupervised learning perspectives. AL, instead of optimizing the informativeness of an existing training set, it expands the dataset to include the most informative and/or representative observations [7]. It is an iterative process where a supervised model is trained and simultaneously identifies the most informative unlabeled observations to increase the performance of that classifier. The combination of SSL with AL has been explored in the past, achieving state-of-the-art results [8].

Several studies have pointed out the limitations of AL within an Imbalanced Learning context [9]. With imbalanced data, AL approaches frequently have low performance, high computational time, or data annotation costs. Studies addressing this issue tend to adopt classifier-level modifications, such as the Weighted Extreme Learning Machine [9, 10, 11]. However, classifier or query function-level modifications (See Section 2.1) have limited applicability since a universally good AL strategy has not been found [7]. Other methods address imbalance learning by weighing the observations as a function of the observation’s class imbalance ratio [12]. Alternatively, other methods reduce the imbalanced learning bias by combining Informative and Representative-based query approaches (see Section 2.1) [13]. Another approach to deal with imbalanced data and data scarcity, in general, is data augmentation. This approach has the advantage of being classifier-agnostic, potentially reduces the imbalanced learning bias, and also works as a regularization method in

data-scarce environments, such as AL implementations [14]. However, most recent studies improve the AL performance by modifying the design/choice of the classifier and query functions used.

The usage of data augmentation in AL is not new. The literature found on the topic (see Section 3.1) focuses on either image classification or Natural Language Processing and uses Deep Learning-based data augmentation to improve the performance of neural network architectures in AL. These methods, although showing promising results, represent a limited perspective of the potential of data augmentation in a real-world setting. First, using Deep Learning in an iterative setting requires access to significant computational power. Second, these models tend to use sophisticated data augmentation methods, whose implementation may not be accessible to the non-sophisticated user. Third, the studies found on the topic are specific to the domain, classifier and data augmentation method. Consequently, the direct effect of data augmentation is unclear: these studies implement different neural network-based techniques for different classification problems, whose performance may be attributed to various elements within the AL framework.

In this study, we explore the effect of data augmentation in AL in a context-agnostic setting, along with two different data augmentation policies: oversampling (where the amount of data generated for each class equals the amount of data belonging to the majority class) and non-constant data augmentation policies (where the amount of data generated exceeds the amount of data belonging to the majority class in varying quantities) between iterations. We start by conceptualizing the AL framework and each of its elements, as well as the modifications involved to implement data augmentation in the AL iterative process. We argue that simple, non-domain specific data augmentation heuristics are sufficient to improve the performance of AL implementations, without the need to resort to deep learning-based data augmentation algorithms.

When compared to the standard AL framework, the proposed framework contains two additional components: the Generator and the Hyperparameter Optimizer. We implement a modified version of Geometric Synthetic Minority Oversampling Technique (G-SMOTE) [15] as a data augmentation method with an optimized generation policy (explained in Section 3). The hyperparameter optimization module is used to find the best data augmentation policy at each iteration. We test the effectiveness of the proposed method in 10 datasets of different domains. We implement 3 AL frameworks

(standard, **oversampling** and varying data augmentation) using 4 different classifiers, 3 different performance metrics and calculate 2 AL-specific performance metrics.

The rest of this manuscript is structured as follows: Section 2.1 describes the state-of-the-art in AL. Section 3 describes the state-of-the-art in Data Augmentation. Section 4 describes the proposed method. Section 5 describes the methodology of the study’s experiment. Section 6 presents the results obtained from the experiment, as well as a discussion of these results. Section 7 presents the conclusions drawn from this study.

## 2. Background

### 2.1. Active Learning

This paper focuses on pool-based AL methods as defined in [16]. The goal of AL models is to maximize the performance of a classifier,  $f_c$ , while annotating as least observations,  $x_i$ , as possible. They use a data pool,  $\mathcal{D}$ , where  $\mathcal{D} = \mathcal{D}_{lab} \cup \mathcal{D}_{pool}$  and  $|\mathcal{D}_{pool}| \gg |\mathcal{D}_{lab}|$ .  $\mathcal{D}_{pool}$  and  $\mathcal{D}_{lab}$  refer to the sets of unlabeled and labeled data, respectively. Having a budget of  $T$  iterations (where  $t = 1, 2, \dots, T$ ) and  $n$  annotations per iteration, at iteration  $t$ ,  $f_c$  is trained using  $\mathcal{D}_{lab}^t$  to produce, for each  $x_i \in \mathcal{D}_{pool}^t$ , an uncertainty score using an acquisition function  $f_{acq}(x_i; f_c)$ . These uncertainty scores are used to annotate the  $n$  observations with highest uncertainty from  $\mathcal{D}_{pool}^t$  to form  $\mathcal{D}_{new}^t$ . The iteration ends with the update of  $\mathcal{D}_{lab}^{t+1} = \mathcal{D}_{lab}^t \cup \mathcal{D}_{new}^t$  and  $\mathcal{D}_{pool}^{t+1} = \mathcal{D}_{pool}^t \setminus \mathcal{D}_{new}^t$  [17, 2]. This process is shown in Figure 1.

Before the start of the iterative process, assuming  $\mathcal{D}_{lab}^{t=0} = \emptyset$ , the data used to populate  $\mathcal{D}_{lab}^{t=1}$  is typically collected randomly from  $\mathcal{D} = \mathcal{D}_{pool}^{t=0}$  and is labeled by a supervisor [18, 19, 20].

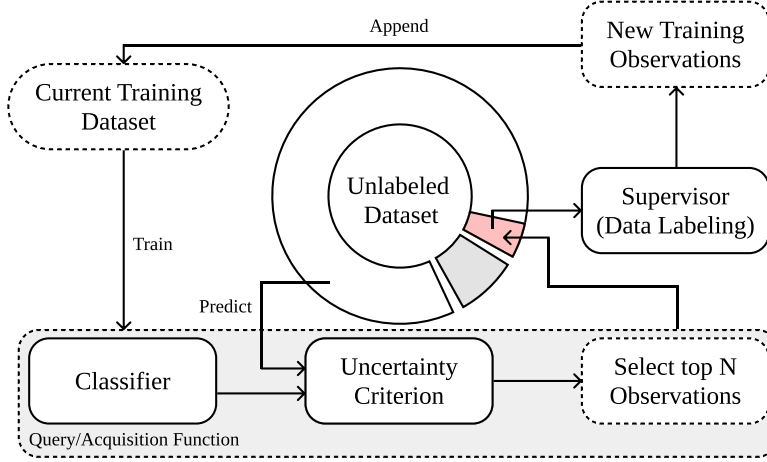


Figure 1: Diagram depicting a typical AL iteration. In the first iteration, the training set collected during the initialization process becomes the “Current Training Dataset”.

107 Research focused on AL has typically been focused on the specification of  
 108  $f_{acq}$  and domain-specific applications. Acquisition functions can be divided  
 109 into two different categories [21, 22]:

- 110 1. Informative-based query strategies. These strategies use the classifier’s  
 111 output to assess the importance of each observation towards the perfor-  
 112 mance of the classifier. These strategies focus on quantifying the class  
 113 uncertainty of the unlabeled observations. Since these techniques do  
 114 not account for the relationships between the unlabeled observations  
 115 and treats each observation independently [23].
- 116 2. Representative-based query strategies. These strategies estimate the  
 117 optimal set of observations that will optimize the classifier’s perfor-  
 118 mance. This strategy contains 3 main approaches: Density-based,  
 119 Diversity-based and Exploration of graph structures. Although this  
 120 method addresses the problem of sampling bias and redundant instance  
 121 selection, these strategies typically require more observations in order  
 122 to reach the desired classification performance [22].

123 Although there are significant contributions towards the development of  
 124 more robust query functions and classifiers in AL, modifications to AL’s

125 basic structure is rarely explored. In [19] the authors introduce a loss predic-  
126 tion module in the AL framework to replace the uncertainty criterion. This  
127 model implements a second classifier to predict the expected loss of the un-  
128 labeled observations (using the actual losses collected during the training of  
129 the original classifier) and return the unlabeled observations with the high-  
130 est expected loss. However, this contribution is specific to neural networks  
131 (and more specifically, to deep neural networks) and was only tested for image  
132 classification.



133 In [5] the authors propose the usage of semi-supervised learning during  
 134 both the initialization of the AL and the iterative process as well. However,  
 135 this method was proposed specifically for deep learning applications. In [18],  
 136 the authors introduce the generator element in the AL framework (discussed  
 137 in Section 4) using an oversampling method, showing that this method ef-  
 138 fectively addresses the limitations of imbalanced learning. However, this  
 139 method was implemented specifically in the Remote Sensing domain and  
 140 used an oversampling strategy without consideration for the actual amount  
 141 of artificial data generated, which may limit its performance.

## 142 2.2. Query Strategies

143 A query strategy/function encompasses all the steps prior to the data  
 144 labeling within an AL iteration. They focus on finding the observations’  
 145 informativeness, representativeness or both [21, 22]. Representative query  
 146 strategies are generally less efficient in data selection than Informative query  
 147 strategies [22]. However, recent research often use representative approaches  
 148 alongside informative approaches [21, 24]. Representative query strategies  
 149 are explored via 3 main approaches [22]:

- 150 1. Density-based, which select representative observations from high den-  
 151 sity regions. [25, 3, 26] used a density-based approach using clustering  
 152 algorithms to select the observations closest to the centroid of each  
 153 cluster.
- 154 2. Diversity-based, which select the N observations at each iteration that  
 155 maximize the diversity in the training data. The diversity-based ap-  
 156 proach was developed to avoid the selection of redundant observations  
 157 in batch-mode learning [27].
- 158 3. Graph-based, which find the most representative nodes and edges of a  
 159 graph network [28]. Since these methods are specific to graph network  
 160 data, they have a more limited applicability.

161 Informative query strategies, unlike representative query strategies, do  
 162 not account for the structure of the unlabeled dataset. As a result, this type  
 163 of strategy may lead to the inefficient selection of observations (*i.e.*, redun-  
 164 dant observations with similar profiles) [22]. Research on more robust selec-  
 165 tion criteria attempts to address the efficiency problem. This is motivated by

166 the importance of the selection criteria in AL’s iterative process [29]. Specifi-  
 167 cally, Settles [30] observed that in some datasets informative query strategies  
 168 fail to outperform the random selection of observations. Generally, the Ran-  
 169 dom Selection query method is used as a baseline. This method disregards  
 170 the class membership probabilities produced by the classifier and returns  $N$   
 171 random points from the dataset without following any specific criteria.

172 A frequently used query strategy is Uncertainty Sampling, originally pro-  
 173 posed in [31]. Using this method, the estimation of an observation’s uncer-  
 174 tainty is based on the target class with the highest probability ( $p_a$ , according  
 175 to the classifier) and the uncertainty is calculated as  $1 - p_a$ . However, since  
 176 this method dismissed the classifier’s predictions on the remaining labels, the  
 177 Breaking Ties criterion was proposed to address this limitation for multiclass  
 178 problems [32]. This method uses the two target classes with highest probabil-  
 179 ity ( $p_a$  and  $p_b$ , according to the classifier) and the uncertainty is calculated as  
 180  $p_a - p_b$  (in this case, the lower the output value, the higher the uncertainty).  
 181 Recent variants of the Breaking Ties criterion, such as the Modified Breaking  
 182 Ties, attempted to fix some limitations of the original method [33, 34].

183 Another common informative query strategy is the calculation of Shan-  
 184 non’s Entropy. This metric measures the level of uncertainty based on the  
 185 probabilities of a set of possible events. Its formula is given by  $H(p) =$   
 186  $-\sum_{i=0}^n p_i \log_2 p_i$ , having  $p$  as the set of probabilities of all target classes.  
 187 The application of the Entropy uncertainty criterion is also frequently ap-  
 188 plied in Deep Active Learning [20]. Other Entropy-based methods were also  
 189 developed for more specific applications. For example, an ensemble querying  
 190 approach known as Entropy Querying-by-Bagging uses the predictions of all  
 191 estimators to find the maximum entropy of each observation [35].

192 The Query by Committee (QBC) strategy was developed to address en-  
 193 semble classifiers. It is a disagreement based strategy attempts to maxi-  
 194 mize the information gain at each iteration by computing the disagreement  
 195 of the predictions over the estimators that form the ensemble. The En-  
 196 tropy Querying-by-Bagging and Query-by-Boosting methods are also ensem-  
 197 ble strategies. Query by boosting and bagging methods were found to achieve  
 198 a good performance over various datasets [36], while the performance between  
 199 the two strategies appears to differ significantly across various scenarios [37].

200 Other classifier-specific query strategies were also developed for different  
 201 applications. However, these methods have the disadvantage of depending  
 202 on the classifier being used. For example, Margin Sampling is a well studied  
 203 strategy that uses a Support Vector Machine as its classifier in order to select

the unlabeled observations closest to its decision boundaries [22]. Although, since this method is known to lead to the excessive selection of observations in dense regions [38], it was improved in various ways. In [38] the authors extend this strategy by applying the manifold-preserving graph reduction algorithm beyond the normal Margin Sampling method.

### 3. Data Augmentation Methods

Data Augmentation methods expand the training dataset by introducing new and informative observations [39]. The production of artificial data may be done via the introduction of perturbations on the input [40], feature [41] or output space [39]. Data Augmentation methods may be divided into Heuristic and Neural Network-based approaches [42]. In addition, they may also be distinguished based on its data generation policy, whether local (considers a local/specific subset of the dataset) or global (considers the overall distribution of the training dataset). Figure 2 shows the general taxonomy of Heuristic Data Augmentation methods. Finding the appropriate Data Augmentation method generally depends on the domain [41], although some studies discuss which methods are more appropriate according to the domain [42, 43, 44].

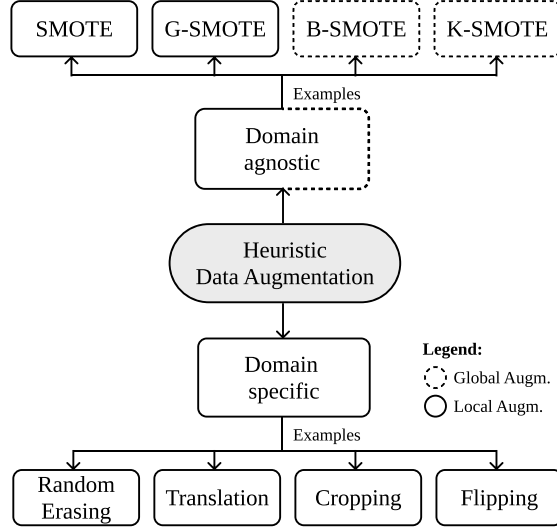


Figure 2: Schema containing a general Heuristic Data Augmentation taxonomy.

Heuristic approaches attempt to generate new and relevant observations through the application a predefined procedure, usually incorporating some degree of randomness [45]. Since these methods typically occur in the input space, they require less data and computational power when compared to Neural Network methods. Neural Network approaches, on the other hand, map the original input space into a lower-dimensional representation, known as the feature space [41]. The generation of artificial data occurs in the feature space and is reconstructed into the input space. Although these methods allow the generation of less noisy data in high-dimensional contexts and more plausible artificial data, they are significantly more computationally intensive. Considering the scope of this paper (the paper’s contribution is described in Sections 1 and 4), the computational power available for this experiment and the breadth of datasets used in our experimental procedure, we will focus on domain-agnostic heuristic data augmentation methods.

While some techniques may depend on the domain, others are domain-agnostic. For example, Random Erasing [40], Translation, Cropping and Flipping are image data-specific augmentation methods. Other methods, such as most of the variants of the Synthetic Minority Oversampling TEchnique (SMOTE) [46], may be considered domain agnostic. However, SMOTE methods were originally developed as oversamplers, whose goal is to balance

243 the class frequencies of the target variable in the training dataset and ad-  
 244 dress the class imbalance bias [47]. Therefore, oversampling methods may be  
 245 considered a subset of Data Augmentation. Data Augmentation strategies  
 246 may follow varying augmentation strategies, which does not necessarily de-  
 247 pend on the target class distribution. An example of the differences among  
 248 general data augmentation and oversampling generation strategies is shown  
 249 in Figure 3.

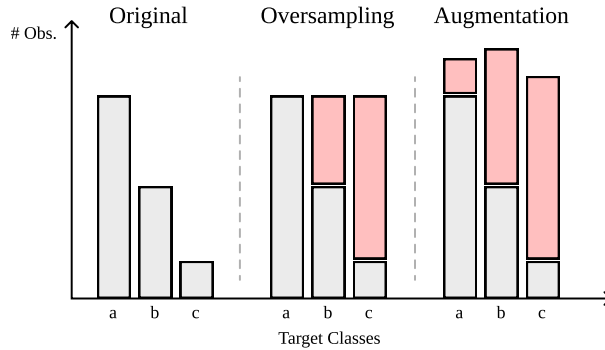


Figure 3: Examples of data augmentation Strategies. The salmon-colored bars represent artificial data using the normal oversampling (center group) and an example of augmentation (right group) strategies.

250 The simplest approach found in the literature is randomly duplicating  
 251 existing training observations. As a non-informed data generation method,  
 252 although simple to implement, it increases the risk of overfitting and generally  
 253 performs worse than other informed heuristic methods [48].

254 The SMOTE method generates artificial data via the linear interpolation  
 255 between a random observation and one of its  $k$ -nearest neighbors (also ran-  
 256 domly selected) [46]. Although simple and effective, it also contains several  
 257 limitations which motivated the development other variants, discussed below.  
 258 Specifically, its selection mechanism does not consider the global structure of  
 259 the dataset while its generation mechanism introduces little variability into  
 260 the training dataset [15]. Borderline-SMOTE (B-SMOTE) [49] improves the  
 261 selection mechanism by attributing a larger importance to the observations  
 262 closer to the decision boundaries. The selected observations are used to run  
 263 the SMOTE method in order to produce better defined decision boundaries.  
 264 A more recent improvement of the selection mechanism is K-means SMOTE

265 (K-SMOTE) [50]. This method uses a clustering-based approach to over-  
 266 come imbalances between and within classes, while considering the densities  
 267 of each region of the input space.

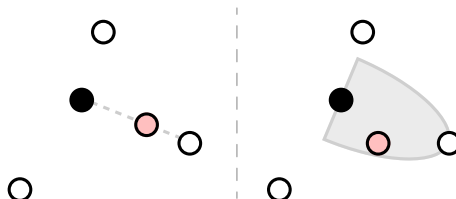


Figure 4: Examples of data generation using SMOTE and G-SMOTE. In this example, both G-SMOTE’s deformation and truncation parameters assume values around 0.5.

268 G-SMOTE [15] modifies SMOTE’s generation mechanism. Instead of  
 269 generating an observation as a linear combination between 2 others, it gen-  
 270 erates observations within an hypersphere defined using the selected obser-  
 271 vation as its center and one of its nearest neighbors as its boundary. The  
 272 hypersphere contains two hyperparameters, the truncation and deformation  
 273 factors, which limit the area of the hypersphere. The difference between  
 274 SMOTE and G-SMOTE is shown in Figure 4. Reference [48] found that  
 275 G-SMOTE outperforms various state-of-the-art oversamplers.

### 276 3.1. Data Augmentation in Active Learning

277  
 278 As found in Section 2.1, the improvement of the AL framework found in  
 279 the literature are mostly focused on modifications of the classifier or query  
 280 strategy. However, a few recent AL applications implementing data aug-  
 281 mentation were found. The method proposed in [14], Look-Ahead Data  
 282 Acquisition for Deep Active Learning, implement image data specific data  
 283 augmentation to train a deep learning classifier. However, in this study,  
 284 data augmentation is based on the unlabeled observations and occurs before  
 285 the unlabeled data selection. In [51] the proposed AL method was designed  
 286 specifically for image data classification, where a deep learning model was  
 287 implemented as a classifier, but its architecture is not described. Other AL  
 288 frameworks implementing data augmentation may also be found for Natural  
 289 Language Processing applications [52, 53]. However, these methods were de-

signed for specific within that domain and are not necessarily transferrable to other domains or tasks.

## 4. Proposed Method

Based on the literature found on AL, most of the contributions and novel implementations of AL algorithms focused on the improvement of the choice/architecture of the classifier or the improvement of the uncertainty criterion. In addition, the resulting classification performance of AL-trained classifiers is frequently inconsistent and marginally improve the classification performance when compared to classifiers trained over the full training set. Finally, in [18] the authors also found a significant variability of the data selection efficiency during different runs of the AL iterative process. In that study the authors proposed a new element within the AL framework, the generator, which was able to marginally reduce the variability previously identified. However, this modification was applied in a Land Use/Land Cover context which contains specific characteristics that are not necessarily found in other supervised learning problems. Specifically, these types of datasets are high dimensional and have limited data variability within each class (*i.e.*, cohesive spectral signatures within classes) due to their geographical proximity. Furthermore, the implementation of the generator was done using a simple oversampling augmentation policy, which limits the possibility of employing other techniques with an undefined target amount of data generated at each iteration.

This paper provides a context-agnostic AL framework towards the integration of Data Augmentation within AL, with the following contributions:

1. Improvement of the AL framework by introducing a parameter tuning stage using only the labeled dataset available at the current iteration (*i.e.*, no labeled hold-out set is needed).
2. Generalization of the generator module proposed in [18] from oversampling techniques to any other data augmentation mechanism and/or policy.
3. Implementation of data augmentation outside of the Deep AL realm, which was not previously found in the literature.





344 which represents a percentage of the majority class  $C_{maj}$  (*e.g.*, an augmenta-  
 345 tion factor of 1.2 will ensure there are  $\text{count}(C_{maj}) \times 1.2$  observations in every  
 346 class). In this paper’s experiment, the data generation mechanism is similar  
 347 to the one in [18]. This allows the direct comparison of the two frameworks  
 348 and establish a causality of the performance variations to the data generation  
 349 mechanism (*i.e.*, augmentation vs normal oversampling) and hyperparameter  
 350 tuning steps.

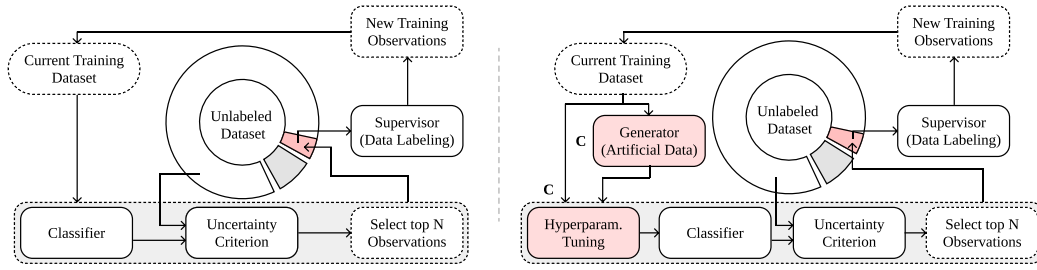


Figure 6: Simplified diagrams highlighting the differences between the proposed and standard AL iterations. The proposed modifications are highlight in red and marked with a boldface “C”.

351 The comparison of diagrams between the proposed and standard AL  
 352 frameworks is shown in Figure 6. In the proposed framework, we (1) gen-  
 353 eralize the generator module to accept any data augmentation method or  
 354 policy and (2) a hyperparameter tuning module to estimate the optimal  
 355 data augmentation policy. This framework was designed to be task-agnostic.  
 356 Specifically, any data augmentation method (domain specific or not) may be  
 357 used, as well as any other parameter search method. It is also expected to  
 358 be compatible with other AL modifications, including the ones that do not  
 359 affect solely the classifier or uncertainty criterion, such as the one proposed  
 360 in [19].

## 361 5. Methodology

362  
 363 This section describes the different elements included in the experimental  
 364 procedure. The datasets used were acquired in open data repositories and its  
 365 sources and preprocessing steps are defined in Subsection 5.1. The choice of  
 366 classifiers used in the experiment are defined in Subsection 5.2. The metrics

367 chosen to measure AL performance and overall classification performance  
 368 are defined in Subsection 5.3. The experimental procedure is described in  
 369 Subsection 5.4. The implementation of the experiment and resources used  
 370 to do so are described in Subsection 5.5.

371 The methodology developed serves 2 purposes: (1) Compare classification  
 372 performance once all the AL procedures are completed (*i.e.*, optimal perfor-  
 373 mance of a classifier trained via iterative data selection) and (2) Compare the  
 374 amount of data required to reach specific performance thresholds (*i.e.*, num-  
 375 ber of AL iterations required to reach similar classification performances).

### 376 5.1. Datasets

377  
 378 The datasets used to test the proposed method are publicly available in  
 379 open data repositories. Specifically, they were retrieved from OpenML and  
 380 the UCI Machine Learning Repository. They were chosen considering dif-  
 381 ferent domains of application, imbalance ratios, dimensionality and number  
 382 of target classes, all of them focused on classification tasks. The goal is to  
 383 demonstrate the performance of the different AL frameworks in various sce-  
 384 narios and domains. The data preprocessing approach was similar across all  
 385 datasets. Table 1 describes the key properties of the 10 preprocessed datasets  
 386 where the experimental procedure was applied.

Dataset	Features	Instances	Minority instances	Majority instances	IR	Classes
Image Segmentation	14	1155	165	165	1.0	7
Mfeat Zernike	47	1994	198	200	1.01	10
Texture	40	1824	165	166	1.01	11
Waveform	40	1666	551	564	1.02	3
Pendigits	16	1832	176	191	1.09	10
Vehicle	18	846	199	218	1.1	4
Mice Protein	69	1073	105	150	1.43	8
Gas Drift	128	1987	234	430	1.84	6
Japanese Vowels	12	1992	156	323	2.07	9
Baseball	15	1320	57	1196	20.98	3

Table 1: Description of the datasets collected after data preprocessing. The sampling strategy is similar across datasets. Legend: (IR) Imbalance Ratio

387 The data preprocessing pipeline is depicted as a flowchart in Figure 7.

388 The missing values are removed from each dataset by removing the corre-  
 389 sponding observations. This ensures that the input data in the experiment  
 390 is kept as close to its original form as possible. The non-metric features (*i.e.*,  
 391 binary, categorical and ordinal variables) were removed since the application  
 392 of G-SMOTE is limited to continuous and discrete features. The datasets  
 393 containing over 2000 observations were downsampled in order to maintain  
 394 the datasets to a manageable size. The data sampling procedure preserves  
 395 the relative class frequency of the dataset, in order to maintain the Imbal-  
 396 ance Ratio (IR) originally found in each dataset (where  $IR = \frac{\text{count}(C_{maj})}{\text{count}(C_{min})}$ ).  
 397 The remaining features of each dataset are scaled to the range of  $[-1, 1]$  to  
 398 ensure a common range across features.

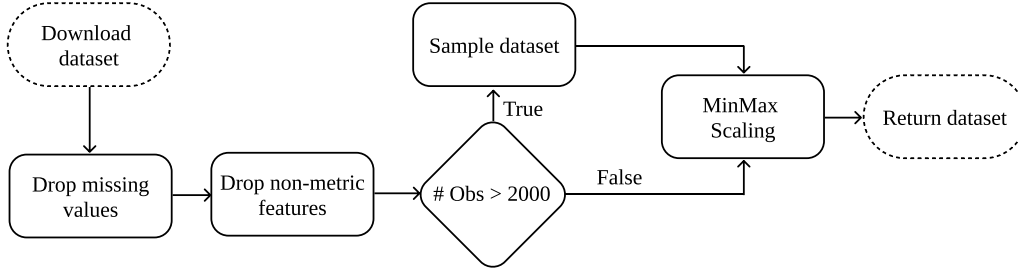


Figure 7: Data preprocessing pipeline.

399 The preprocessed datasets were stored into a SQLite database file and is  
 400 available along with the experiment’s source code in the GitHub repository  
 401 of the project (see Subsection 5.5).

## 402 5.2. Machine Learning Algorithms

403  
 404 We used a total of 4 classification algorithms and a heuristic data aug-  
 405 mentation mechanism. The choice of classifiers was based on the popularity  
 406 and family of the classifiers (tree-based, nearest neighbors-based, ensemble-  
 407 based and linear models). Our proposed method was tested using a Decision  
 408 Tree (DT) [54], a K-nearest neighbors classifier (KNN) [55], a Random For-  
 409 est Classifier (RF) [56] and a Logistic Regression (LR) [57]. Since the target  
 410 variables are multi-class, the LR classifier was implemented using the one-  
 411 versus-all approach. The predicted class is assigned to the label with the  
 412 highest likelihood.

The oversampler G-SMOTE was used as a data augmentation method. The typical data generation policy of oversampling methods is to generate artificial observations on non-majority classes such that the number of majority class observations matches those of each non-majority class. We modified this data generation policy to generate observations for all classes, as a percentage of the number of observations in the majority class. In addition, the original G-SMOTE algorithm was modified to accept data selection probabilities based on classification uncertainty. These modifications are discussed in Section 4.

Every AL procedure was tested with different selection criteria: Random Selection, Entropy and Breaking Ties. The baseline used is the standard AL procedure. As a benchmark, we add the AL procedure using G-SMOTE as a normal oversampling method, as proposed in [18]. Our proposed method was implemented using G-SMOTE as a data augmentation method to generate artificial observations for all classes, while still balancing the class distribution, as described in Section 4.

### 5.3. Evaluation Metrics

Considering the imbalanced nature of the datasets used in the experiment, commonly used performance metrics such as Overall Accuracy (OA), although being intuitive to interpret, are insufficient quantify a model’s classification performance [58]. The Cohen’s Kappa performance metric, similar to OA, is also biased towards high frequency classes since its definition is closely related to the OA metric, making its behavior consistent with OA [59]. However, these metrics remain popular choices for the evaluation of classification performance. Other performance metrics like  $Precision = \frac{TP}{TP+FN}$ ,  $Recall = \frac{TP}{TP+FN}$  or  $Specificity = \frac{TN}{TN+FP}$  are calculated as a function of True/False Positives (TP and FP) and True/False Negatives (TN and FN) and can be used at a per-class basis instead. In a multiple dataset scenario with varying amount of target classes and meanings, comparing the performance of different models using these metrics becomes impractical.

Based on the recommendations found in [58, 60], we used 2 metrics found to be less sensitive to the class imbalance bias, along with OA as a reference for easier interpretability:

- The Geometric-mean scorer (G-mean) consists of the geometric mean of Specificity and Recall [60]. Both metrics are calculated in a multiclass

449 context considering a one-versus-all approach. For multiclass problems,  
 450 the G-mean scorer is calculated as its average per class values:

$$G\text{-mean} = \sqrt{\overline{Sensitivity} \times \overline{Specificity}}$$

- 451 • The F-score metric consists of the harmonic mean of Precision and  
 452 Recall. The two metrics are also calculated considering a one-versus-  
 453 all approach. The F-score for the multi-class case can be calculated  
 454 using its average per class values [58]:

$$F\text{-score} = 2 \times \frac{\overline{Precision} \times \overline{Recall}}{\overline{Precision} + \overline{Recall}}$$

- 455 • The OA consists of the number of TP divided by the total amount of  
 456 observations. Considering  $c$  as the label for the different classes present  
 457 in a target class, OA is given by the following formula:

$$OA = \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c)}$$

458 The comparison of the performance of AL frameworks is based on its  
 459 data selection and augmentation efficacy. Specifically, an efficient data se-  
 460 lection/generation policy allows the production of classifiers with high per-  
 461 formance on unseen data while using as least non-artificial training data as  
 462 possible. To measure the performance of the different AL setups, we follow  
 463 the recommendations found in [61]. The performance of an AL setup will be  
 464 compared using two AL-specific performance metrics:

- 465 • Area Under the Learning Curve (AULC). It is the sum of the classifi-  
 466 cation performance over a validation/test set of the classifiers trained  
 467 of all AL iterations. To facilitate the interpretability of this metric,  
 468 the resulting AULC scores are fixed within the range  $[0, 1]$  by dividing  
 469 the AULC scores by the total amount of iterations (*i.e.*, the maximum  
 470 performance area).
- 471 • Data Utilization Rate (DUR) [62]. Measures the percentage of training  
 472 data required to reach a given performance threshold, as a ratio of

the percentage of training data required by the baseline framework. This metric is also presented as a percentage of the total amount of training data, without making it relative to the baseline framework. The DUR metric is measured at 45 different performance thresholds, ranging between  $[0.10, 1.00]$  at a 0.02 step.

#### 5.4. Experimental Procedure

The evaluation of different active learners in a live setting is generally expensive, time-consuming and prone to human error. Instead, a common practice is to compare them in an offline environment using labeled datasets [63]. In this scenario, since the dataset is already labeled, the annotation process is done at zero cost. Figure 8 depicts the experiment designed for one dataset over a single run.

A single run starts with the splitting of a preprocessed dataset in 5 different partitions, stratified according to the class frequencies of the target variable using the K-fold Cross Validation method. During this run, an active learner or classifier is trained 5 times using a different partition as the Test set each time. For each training process, a Validation set containing 25% of the subset is created and is used to measure the data selection efficiency (*i.e.*, AULC and DUR using the classification performance metrics, specific to AL). Therefore, for a single training procedure, 20% of the original dataset is used as the Validation set, 20% is used as the Test set and 60% is used as the Train set. The AL simulations and the classifiers' training occur within the Train set. However, the classifiers used to find the maximum performance classification scores are trained over the full Train set. The AL simulations are run over a maximum of 50 iterations (including the initialization step), adding 1.6% of the training set each time (*i.e.*, all AL simulations use less than 80% of the Train set). Once the training phase is completed, the Test set classification scores are calculated using the trained classifiers. For the case of AL, the classifier with the optimal Validation set score is used to estimate the AL's optimal classification performance over unseen data.

The process shown in Figure 8 is repeated over 3 runs using different random seeds over the 10 different datasets collected. The final scores of each AL configuration and classifier correspond to the average of the 3 runs and 5-fold Cross Validation estimations (*i.e.*, the mean score of 15 fits, across 10 datasets).

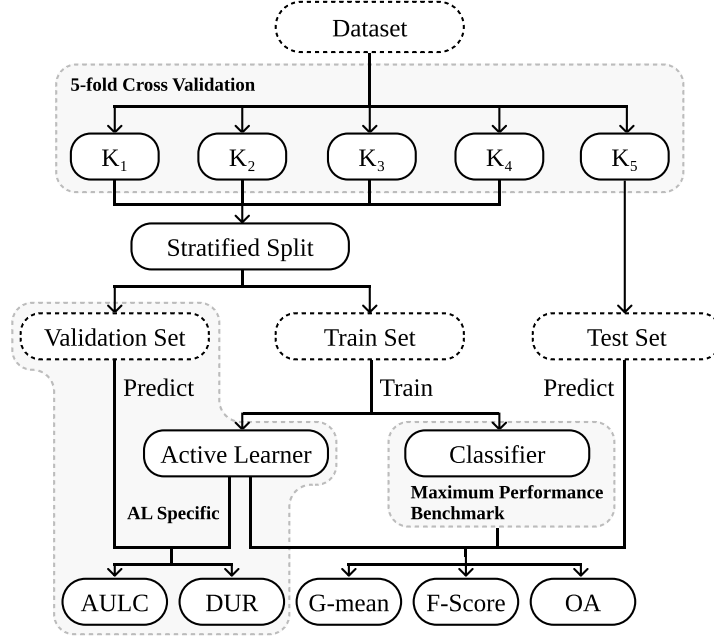


Figure 8: Experimental procedure flowchart. The preprocessed datasets are split into five folds. One of the folds is used to test the best found classifiers using AL and the classifiers trained using the entire training dataset (containing the remaining folds). The training set is used to run both the AL simulations as well as train the normal classifiers. The validation set is used to measure AL-specific performance metrics over each iteration. We use different subsets for overall classification performance and AL-specific performance to avoid data leakage.

509 The hyperparameters defined for the AL frameworks, Classifiers and Gen-  
510 erators are shown in Table 2. In the Generators table, we distinguish the  
511 G-SMOTE algorithm working as a normal oversampling method from G-  
512 SMOTE-AUGM, which performs generates additional artificial data on top  
513 of the usual oversampling mechanism. Since the G-SMOTE-AUGM method  
514 is intended to be used with varying parameter values (via within-iteration  
515 parameter tuning), the parameters were defined as a list of various possible  
516 values.

Active Learners	Hyperparameters	Inputs
Standard	# initial obs.	1.6%
	# additional obs. per iteration	1.6%
	max. iterations + initialization	50
	evaluation metrics	G-mean, F-score, OA
	selection strategy	Random, Entropy, Breaking Ties
	within-iteration param. tuning	None
	generator	None
	classifier	DT, LR, KNN, RF
Oversampling	generator	G-SMOTE
Proposed	generator	G-SMOTE-AUGM
	within-iteration param. tuning	Grid Search K-fold CV
Classifier		
DT	min. samples split	2
	criterion	gini
LR	maximum iterations	100
	multi class	One-vs-All
	solver	liblinear
KNN	penalty	L2 (Ridge)
	# neighbors	5
	weights	uniform
	metric	euclidean
RF	min. samples split	2
	# estimators	100
	criterion	gini
Generator		
G-SMOTE	# neighbors	4
	deformation factor	0.5
	truncation factor	0.5
G-SMOTE-AUGM	# neighbors	3, 4, 5
	deformation factor	0.5
	truncation factor	0.5
	augmentation factor	[1.1, 2.0] at 0.1 step

Table 2: Hyperparameter definition for the active learners, classifiers and generators used in the experiment.



## 517 5.5. *Software Implementation*

518

519 The experiment was implemented using the Python programming lan-  
520 guage, along with the Python libraries Scikit-Learn [64], Imbalanced-Learn [65],  
521 Geometric-SMOTE [15], Research-Learn and ML-Research libraries. All  
522 functions, algorithms, experiments and results are provided in the GitHub  
523 repository of the project.

## 524 6. **Results & Discussion**

525

526 In a multiple dataset experiment, the analysis of results should not rely  
527 uniquely on the average performance scores across datasets. The domain of  
528 application and fluctuations of performance scores between datasets make  
529 the analysis of these averaged results less accurate. Instead, it is generally  
530 recommended the use of the mean ranking scores to extend the analysis [66].  
531 Since mean performance scores are still intuitive to interpret, we will present  
532 and discuss both results. The rank values are assigned based on the mean  
533 scores of 3 different runs of 5-fold Cross Validation (15 performance estima-  
534 tions per dataset) for each combination of dataset, AL configuration, classifier  
535 and performance metric.

### 536 6.1. *Results*

537

538 The average ranking of the AULC estimations of AL methods are shown  
539 in Table 3. The proposed method almost always improves AL performance  
540 and ensures higher data selection efficiency.

Classifier	Evaluation Metric	Standard	Oversampling	Proposed
DT	Accuracy	$2.50 \pm 0.81$	$2.20 \pm 0.40$	<b><math>1.30 \pm 0.64</math></b>
DT	F-score	$2.50 \pm 0.81$	$2.10 \pm 0.30$	<b><math>1.40 \pm 0.80</math></b>
DT	G-mean	$2.70 \pm 0.64$	$2.00 \pm 0.45$	<b><math>1.30 \pm 0.64</math></b>
KNN	Accuracy	$2.40 \pm 0.80$	$1.90 \pm 0.54$	<b><math>1.70 \pm 0.90</math></b>
KNN	F-score	$2.60 \pm 0.66$	$1.80 \pm 0.40$	<b><math>1.60 \pm 0.92</math></b>
KNN	G-mean	$2.80 \pm 0.40$	$1.70 \pm 0.46$	<b><math>1.50 \pm 0.81</math></b>
LR	Accuracy	$2.60 \pm 0.66$	$2.10 \pm 0.54$	<b><math>1.30 \pm 0.64</math></b>
LR	F-score	$2.80 \pm 0.40$	$2.00 \pm 0.45$	<b><math>1.20 \pm 0.60</math></b>
LR	G-mean	$2.80 \pm 0.40$	$2.00 \pm 0.45$	<b><math>1.20 \pm 0.60</math></b>
RF	Accuracy	$2.60 \pm 0.66$	$1.90 \pm 0.54$	<b><math>1.50 \pm 0.81</math></b>
RF	F-score	$2.60 \pm 0.66$	$2.00 \pm 0.45$	<b><math>1.40 \pm 0.80</math></b>
RF	G-mean	$2.80 \pm 0.40$	<b><math>1.60 \pm 0.49</math></b>	<b><math>1.60 \pm 0.80</math></b>

Table 3: Mean rankings of the AULC metric over the different datasets (10), folds (5) and runs (3) used in the experiment. The proposed method always improves the results of the original framework and on average almost always improves the results of the oversampling framework.

541 Table 4 shows the average AULC scores, grouped by classifier, Evalua-  
542 tion Metric and AL framework. The variation in performance across active  
543 learners is consistent with the mean rankings found in Table 3, while showing  
544 significant AULC score differences between the proposed AL method and the  
545 oversampling AL method.

546 The average DUR scores were calculated for various G-mean thresholds,  
547 varying between 0.1 and 1.0 at a 0.02 step (45 different thresholds in total).  
548 Table 5 shows the results obtained for these scores starting from a G-mean  
549 score of 0.6 and was filtered to show only the thresholds ending with 0 or 6.  
550 In most cases, the proposed method reduces the amount of data annotation  
551 required to reach each G-mean score threshold.

552 The DUR scores relative to the Standard AL method are shown in Fig-  
553 ure 9. A DUR below 1 means that the Proposed/Oversampling method  
554 requires less data than the Standard AL method to reach the same perfor-  
555 mance threshold. For example, running an AL simulation using the KNN  
556 classifier requires 69.6% of the amount of data required by the Standard AL  
557 method using the same classifier to reach an F-Score of 0.62 (*i.e.*, requires  
558 30.4% less data).

559 The mean optimal classification scores of AL methods and Classifiers

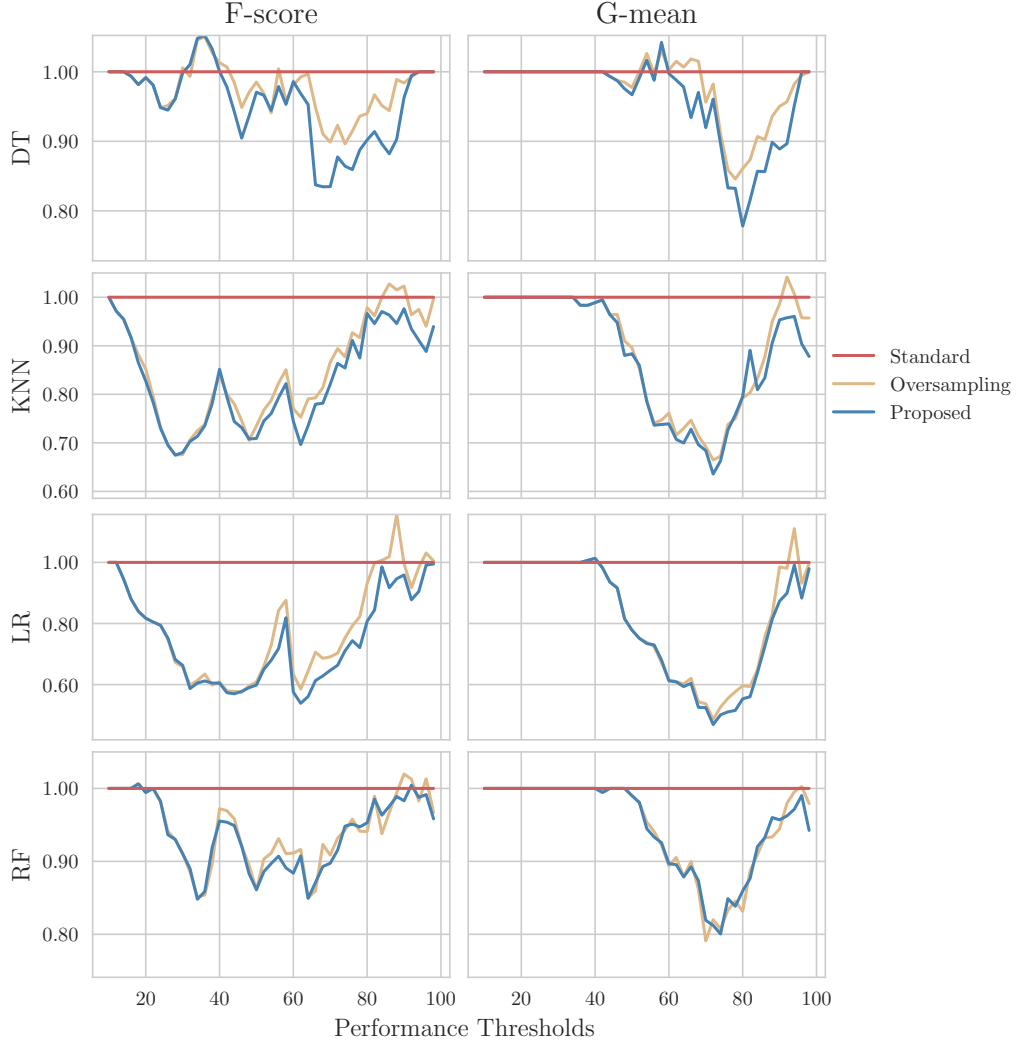


Figure 9: Mean data utilization rates. The y-axis shows the percentage of data (relative to the baseline AL framework) required to reach the different performance thresholds.

Classifier	Evaluation Metric	Standard	Oversampling	Proposed
DT	Accuracy	$0.733 \pm 0.092$	$0.732 \pm 0.087$	<b><math>0.740 \pm 0.087</math></b>
DT	F-score	$0.695 \pm 0.088$	$0.698 \pm 0.090$	<b><math>0.705 \pm 0.092</math></b>
DT	G-mean	$0.804 \pm 0.065$	$0.811 \pm 0.060$	<b><math>0.816 \pm 0.062</math></b>
KNN	Accuracy	$0.816 \pm 0.091$	$0.818 \pm 0.088$	<b><math>0.822 \pm 0.091</math></b>
KNN	F-score	$0.775 \pm 0.102$	$0.784 \pm 0.108$	<b><math>0.788 \pm 0.111</math></b>
KNN	G-mean	$0.852 \pm 0.084$	$0.866 \pm 0.072$	<b><math>0.869 \pm 0.074</math></b>
LR	Accuracy	$0.802 \pm 0.091$	$0.812 \pm 0.088$	<b><math>0.821 \pm 0.086</math></b>
LR	F-score	$0.749 \pm 0.112$	$0.773 \pm 0.116$	<b><math>0.784 \pm 0.115</math></b>
LR	G-mean	$0.839 \pm 0.093$	$0.870 \pm 0.065$	<b><math>0.875 \pm 0.064</math></b>
RF	Accuracy	$0.861 \pm 0.076$	$0.861 \pm 0.075$	<b><math>0.862 \pm 0.077</math></b>
RF	F-score	$0.823 \pm 0.105$	$0.827 \pm 0.105$	<b><math>0.829 \pm 0.105</math></b>
RF	G-mean	$0.886 \pm 0.077$	<b><math>0.895 \pm 0.063</math></b>	<b><math>0.895 \pm 0.065</math></b>

Table 4: Average AULC of each AL configuration tested. Each AULC score is calculated using the performance scores of each iteration in the validation set. By the end of the iterative process, each AL configuration used a maximum of 80% instances of the 60% instances that compose the training sets (*i.e.*, 48% of the entire preprocessed dataset).

(fully labeled training set, without AL) is shown in Table 6. The proposed AL method produces classifiers that are almost always able to outperform classifiers using the full training set (*i.e.*, the ones labeled as MP).

## 6.2. Statistical Analysis

When checking for statistical significance in a multiple dataset context it is important to account for the multiple comparison problem. Consequently, our statistical analysis focuses on the recommendations found in [66]. Overall, we perform 3 statistical tests. The Friedman test [67] is used to understand whether there is a statistically significant difference in performance between the 3 AL frameworks. As post hoc analysis, the Wilcoxon signed-rank test [68] was used to check for statistical significance between the performance of the proposed AL method and the oversampling AL method across datasets. As a second post hoc analysis, the Holm-Bonferroni [69] method was used to check for statistical significance between the methods using data generators and the Standard AL framework across classifiers and evaluation metrics.

G-mean Score	Classifier	Standard	Oversampling	Proposed
0.60	DT	3.2%	<b>3.1%</b>	3.2%
0.60	KNN	3.6%	2.6%	<b>2.5%</b>
0.60	LR	3.9%	<b>2.2%</b>	<b>2.2%</b>
0.60	RF	2.4%	<b>2.1%</b>	<b>2.1%</b>
0.66	DT	4.6%	4.6%	<b>4.2%</b>
0.66	KNN	4.9%	3.7%	<b>3.5%</b>
0.66	LR	5.7%	3.2%	<b>3.1%</b>
0.66	RF	3.0%	2.8%	<b>2.7%</b>
0.70	DT	6.6%	6.1%	<b>5.8%</b>
0.70	KNN	8.5%	5.0%	<b>4.7%</b>
0.70	LR	9.5%	4.6%	<b>4.3%</b>
0.70	RF	4.5%	<b>3.2%</b>	3.3%
0.76	DT	16.5%	13.0%	<b>12.7%</b>
0.76	KNN	17.8%	9.7%	<b>9.0%</b>
0.76	LR	16.6%	10.0%	<b>7.8%</b>
0.76	RF	10.1%	<b>5.5%</b>	<b>5.5%</b>
0.80	DT	36.1%	30.4%	<b>27.1%</b>
0.80	KNN	22.7%	18.0%	<b>17.8%</b>
0.80	LR	25.2%	16.0%	<b>14.2%</b>
0.80	RF	15.5%	<b>9.0%</b>	9.5%
0.86	DT	60.5%	56.7%	<b>54.5%</b>
0.86	KNN	39.9%	<b>37.0%</b>	37.8%
0.86	LR	32.6%	27.5%	<b>27.0%</b>
0.86	RF	28.0%	<b>25.7%</b>	<b>25.7%</b>
0.90	DT	72.5%	70.7%	<b>67.8%</b>
0.90	KNN	49.9%	50.3%	<b>49.3%</b>
0.90	LR	52.5%	53.8%	<b>49.3%</b>
0.90	RF	44.6%	<b>42.6%</b>	43.5%
0.96	DT	100.0%	<b>99.5%</b>	100.0%
0.96	KNN	79.4%	75.6%	<b>71.6%</b>
0.96	LR	87.5%	83.1%	<b>79.8%</b>
0.96	RF	63.6%	64.2%	<b>63.1%</b>

Table 5: Mean data utilization of AL algorithms, as a percentage of the training set.

Classifier	Evaluation Metric	MP	Standard	Oversampling	Proposed
DT	Accuracy	$0.809 \pm 0.086$	$0.802 \pm 0.089$	$0.806 \pm 0.089$	<b><math>0.812 \pm 0.087</math></b>
DT	F-score	$0.774 \pm 0.107$	$0.772 \pm 0.096$	$0.775 \pm 0.101$	<b><math>0.781 \pm 0.103</math></b>
DT	G-mean	$0.853 \pm 0.081$	$0.854 \pm 0.069$	$0.860 \pm 0.067$	<b><math>0.864 \pm 0.068</math></b>
KNN	Accuracy	$0.882 \pm 0.085$	<b><math>0.883 \pm 0.087</math></b>	$0.877 \pm 0.087$	$0.881 \pm 0.093$
KNN	F-score	$0.848 \pm 0.116$	$0.849 \pm 0.115$	$0.847 \pm 0.118$	<b><math>0.852 \pm 0.121</math></b>
KNN	G-mean	$0.896 \pm 0.094$	$0.899 \pm 0.090$	$0.904 \pm 0.078$	<b><math>0.907 \pm 0.080</math></b>
LR	Accuracy	$0.855 \pm 0.074$	<b><math>0.870 \pm 0.073</math></b>	$0.858 \pm 0.077$	<b><math>0.870 \pm 0.076</math></b>
LR	F-score	$0.812 \pm 0.113$	$0.835 \pm 0.105$	$0.825 \pm 0.106$	<b><math>0.838 \pm 0.106</math></b>
LR	G-mean	$0.875 \pm 0.099$	$0.895 \pm 0.075$	$0.899 \pm 0.059$	<b><math>0.907 \pm 0.059</math></b>
RF	Accuracy	$0.897 \pm 0.080$	$0.905 \pm 0.078$	$0.904 \pm 0.078$	<b><math>0.906 \pm 0.077</math></b>
RF	F-score	$0.867 \pm 0.107$	<b><math>0.877 \pm 0.103</math></b>	$0.875 \pm 0.108$	<b><math>0.877 \pm 0.108</math></b>
RF	G-mean	$0.911 \pm 0.081$	$0.917 \pm 0.078$	$0.923 \pm 0.067$	<b><math>0.925 \pm 0.065</math></b>

Table 6: Optimal classification scores. The Maximum Performance (MP) classification scores are calculated using classifiers trained using the entire training set.

577 Table 7 contains the *p-values* obtained with the Friedman test. The  
578 difference in performance across AL frameworks is statistically significant at  
579 a level of  $\alpha = 0.05$  regardless of the classifier or evaluation metric being  
580 considered.

Classifier	Evaluation Metric	p-value	Significance
DT	Accuracy	2.1e-17	True
DT	F-score	2.5e-24	True
DT	G-mean	2.8e-16	True
KNN	Accuracy	1.1e-46	True
KNN	F-score	1.8e-66	True
KNN	G-mean	6.4e-42	True
LR	Accuracy	9.9e-59	True
LR	F-score	2.0e-76	True
LR	G-mean	2.2e-59	True
RF	Accuracy	5.7e-42	True
RF	F-score	4.6e-55	True
RF	G-mean	1.3e-38	True

Table 7: Results for Friedman test. Statistical significance is tested at a level of  $\alpha = 0.05$ . The null hypothesis is that there is no difference in the classification outcome across oversamplers.

Table 8 contains the *p-values* obtained with the Wilcoxon signed-rank test. The proposed method was able to outperform both the standard AL framework, as well as the AL framework using a normal oversampling policy proposed in [18] with statistical significance in 9 out of 10 datasets.

The *p-values* shown in Table 9 refer to the results of the Holm-Bonferroni test. The proposed method’s superior performance was statistically significant for any combination of classifier and evaluation metric. Simultaneously, the proposed method established statistical significance in the 3 scenarios where the oversampling AL method failed to do so.

### 6.3. Discussion

In this paper we study the application of data augmentation methods through the modification of the standard AL framework. This is done to further reduce the amount of labeled data required to produce a reliable classifier, at the expense of artificial data generation.

In Table 3 we found that the proposed method was able to outperform the Standard AL framework in all scenarios. The mean rankings are consistent with the mean AULC scores found in Table 4, while showing significant performance differences between the proposed method and both the standard

Dataset	Oversampling	Standard
Baseball	5.0e-01	3.4e-01
Gas Drift	<b>3.7e-26</b>	<b>4.6e-57</b>
Image Segmentation	<b>9.6e-18</b>	<b>2.1e-44</b>
Japanese Vowels	<b>2.4e-09</b>	<b>1.6e-32</b>
Mfeat Zernike	<b>1.2e-12</b>	<b>9.5e-40</b>
Mice Protein	<b>6.5e-32</b>	<b>1.5e-61</b>
Pendigits	<b>5.0e-18</b>	<b>2.3e-45</b>
Texture	<b>1.5e-22</b>	<b>6.7e-57</b>
Vehicle	<b>7.4e-11</b>	<b>7.9e-13</b>
Waveform	<b>8.9e-08</b>	<b>2.6e-02</b>

Table 8: Adjusted p-values using the Wilcoxon signed-rank method. Bold values are statistically significant at a level of  $\alpha = 0.05$ . The null hypothesis is that the performance of the proposed framework is similar to that of the oversampling or standard framework.

Classifier	Evaluation Metric	Oversampling	Proposed
DT	Accuracy	<b>4.5e-05</b>	<b>1.6e-10</b>
DT	F-score	<b>1.9e-07</b>	<b>2.7e-10</b>
DT	G-mean	<b>2.5e-06</b>	<b>3.1e-09</b>
KNN	Accuracy	5.5e-02	<b>1.1e-05</b>
KNN	F-score	<b>6.7e-11</b>	<b>6.3e-14</b>
KNN	G-mean	<b>8.3e-06</b>	<b>1.3e-07</b>
LR	Accuracy	8.1e-02	<b>3.4e-06</b>
LR	F-score	<b>7.1e-06</b>	<b>2.0e-20</b>
LR	G-mean	<b>2.2e-07</b>	<b>1.1e-11</b>
RF	Accuracy	2.0e-01	<b>2.8e-02</b>
RF	F-score	<b>2.2e-05</b>	<b>8.1e-07</b>
RF	G-mean	<b>2.0e-04</b>	<b>2.0e-04</b>

Table 9: Adjusted p-values using the Holm-Bonferroni method. Bold values are statistically significant at a level of  $\alpha = 0.05$ . The null hypothesis is that the Oversampling or Proposed method does not perform better than the control method (Standard AL framework).



600 and oversampling methods. The Friedman test in Table 7 showed that the  
601 difference in the performance of these AL frameworks is statistically signifi-  
602 cant, regardless of the classifier or performance metric being used.

603 The proposed method showed more consistent data utilization require-  
604 ments to most of the assessed G-mean score thresholds when compared to  
605 the remaining AL methods, as seen in Table 5. For example, to reach a  
606 G-mean Score of 0.9 using the KNN and LR classifiers, the average amount  
607 of data required with the Oversampling AL approach increased when com-  
608 pared to the Standard approach. However, the proposed method was able to  
609 decrease the amount of data required in both situations. The robustness of  
610 the Proposed method is clearer in Figure 9. In most cases, this method was  
611 able outperform the Oversampling method. At the same time, the proposed  
612 method also addresses inconsistencies in situations where the Oversampling  
613 method was unable to outperform the standard method.

614 The statistical analyses found in Tables 8 and 9 showed that the pro-  
615 posed method’s superiority was statistically significant in all datasets except  
616 one (Baseball) and established statistical significance when compared to the  
617 Standard AL method for all combinations of classifier and performance met-  
618 ric, including when the Oversampling AL method failed to do so. These  
619 results show that the Proposed method increased the reliability of the new  
620 AL framework and improved the quality of the final classifier while using less  
621 data.

622 Even though it was not the core purpose of this study, we found that  
623 the method proposed AL approach consistently outperformed the maximum  
624 performance threshold. Specifically, in Table 6, the performance of the classi-  
625 fiers originating from the proposed method was able to outperform classifiers  
626 trained using the full training dataset in all 12 scenarios except one. This  
627 suggests that the selection of a meaningful training subset training dataset  
628 paired with data augmentation not only matches the classification perfor-  
629 mance of ML algorithms, as it also improves them. Even in a setting with  
630 fully labeled training data, the proposed method may be used as preprocess-  
631 ing method to further optimize classification performance.

632 This study discussed the effect of data augmentation within the AL frame-  
633 work, along with the exploration of optimal augmentation methods within  
634 AL iterations. However, the conceptual nature of this study implies some  
635 limitations. Specifically, the large amount of experiments required to test  
636 the method’s efficacy, along with the limited computational power available,  
637 led to a limited exploration of the grid search’s potential. Future work should

focus into understanding how the usage of a more comprehensive parameter tuning approach improves the quality of the AL method. In addition, the proposed method was not able to outperform the standard AL method in 100% of scenarios. The exploration of other, more complex, data augmentation techniques might further improve its performance through the production of more meaningful training observations. Specifically, in this study we assume that all datasets used follow a manifold, allowing the usage of G-SMOTE as a data augmentation approach. However, this method cannot be used into more complex, non-euclidean spaces. In this scenario, the usage of G-SMOTE is not valid and might lead to the production of noisy data. Deep Learning-based data augmentation techniques are able to address this limitation and improve the overall quality of the artificial data being generated. We also found significant standard errors throughout our experimental results (see Subsection 6.1), which is consistent with the findings in [18, 61]. This suggests that the usage of more robust generators did not decrease the standard error of AL performance. Instead, AL’s performance variability is likely dependent on the quality of its initialization.

## 7. Conclusion

The ability of training ML classifiers is usually limited to the availability of labeled data. However, manually labeling data is often expensive, which makes the usage of AL particularly appealing to select the most informative observations and reduce the amount of required labeled data. On the other hand, the introduction of data variability in the training dataset can also be done via data augmentation. However, most, if not all, AL configurations using some form data augmentation are domain and/or task specific. These methods typically explore deep learning approaches on both classification and data augmentation. Consequently, they may not be applicable for other classification tasks or when the available computational power is insufficient.

In this paper, we proposed a domain-agnostic AL framework that implements Data Augmentation and hyperparameter tuning. We found that a heuristic Data Augmentation algorithm is sufficient to improve the data selection efficiency in AL. Specifically, the data augmentation method used almost always increased AL performance, regardless of the target goal (*i.e.*, optimizing classification or data selection efficiency). The usage of data augmentation reduced the **number** of iterations required to train a classifier with

674 a performance as good as (or better than) classifiers trained with the entire  
675 training dataset (*i.e.*, without using AL). In addition, the proposed method  
676 reduced the size of the training dataset, which is expanded with artificial  
677 data.

678 With this AL configuration, data selection in AL iterations aim towards  
679 observations that optimize the quality of the artificial data produced. The  
680 substitution of less informative labeled data with artificial data is especially  
681 useful in this context, since it allows the reduction of some of the user interac-  
682 tion necessary to reach a sufficiently informative dataset. In order to further  
683 improve the proposed method future work will (1) focus on the development  
684 of methods with varying data augmentation policies depending on the differ-  
685 ent input space regions, (2) develop augmentation-sensitive query functions  
686 capable of avoiding the unnecessary selection of similar observations from the  
687 unlabeled dataset and (3) better understand the gap between heuristic/input  
688 space data augmentation techniques and neural network/feature space data  
689 augmentation techniques in an AL context.

## 690 **Declarations**

### 691 *Funding*

692 This research was supported by three research grants of the Portuguese  
693 Foundation for Science and Technology (“Fundação para a Ciência e a Tec-  
694 nologia”), references SFRH/BD/151473/2021, DSAIPA/DS/0116/2019 and  
695 PCIF/SSI/0102/2017.

### 696 *Code availability*

697 The analyses and source code is available at [github.com/joaopfonseca/ml-](https://github.com/joaopfonseca/ml-research)  
698 [research](https://github.com/joaopfonseca/ml-research).

## 699 **References**

- 700 [1] V. Nath, D. Yang, B. A. Landman, D. Xu, H. R. Roth, Diminishing  
701 uncertainty within the training pool: Active learning for medical image  
702 segmentation, *IEEE Transactions on Medical Imaging* 40 (10) (2021)  
703 2534–2547.
- 704 [2] Y. Sverchkov, M. Craven, A review of active learning approaches to  
705 experimental design for uncovering biological networks, *PLoS Compu-*  
706 *tational Biology* 13 (2017) e1005466.

- 707 [3] X. Li, D. Kuang, C. X. Ling, Active learning for hierarchical text classification,  
708 Lecture Notes in Computer Science (including subseries Lecture  
709 Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7301  
710 LNAI (2012) 14–25.
- 711 [4] Y. Li, J. Yin, L. Chen, Seal: Semisupervised adversarial active learning  
712 on attributed graphs, IEEE Transactions on Neural Networks and  
713 Learning Systems 32 (7) (2021) 3136–3147.
- 714 [5] O. Siméoni, M. Budnik, Y. Avrithis, G. Gravier, Rethinking deep active  
715 learning: Using unlabeled data at model training, Proceedings -  
716 International Conference on Pattern Recognition (2020) 1220–1227.
- 717 [6] J. E. Van Engelen, H. H. Hoos, A survey on semi-supervised learning,  
718 Machine Learning 109 (2) (2020) 373–440.
- 719 [7] O. Sener, S. Savarese, Active learning for convolutional neural networks:  
720 A core-set approach, in: International Conference on Learning Representations,  
721 2018.
- 722 [8] Y. Leng, X. Xu, G. Qi, Combining active learning and semi-supervised  
723 learning to construct svm classifier, Knowledge-Based Systems 44 (2013)  
724 121–131.
- 725 [9] H. Yu, X. Yang, S. Zheng, C. Sun, Active learning from imbalanced  
726 data: A solution of online weighted extreme learning machine, IEEE  
727 Transactions on Neural Networks and Learning Systems 30 (2019) 1088–  
728 1103.
- 729 [10] W. Zong, G.-B. Huang, Y. Chen, Weighted extreme learning machine  
730 for imbalance learning, Neurocomputing 101 (2013) 229–242.
- 731 [11] J. Qin, C. Wang, Q. Zou, Y. Sun, B. Chen, Active learning with extreme  
732 learning machine for online imbalanced multiclass classification,  
733 Knowledge-Based Systems 231 (2021) 107385.
- 734 [12] W. Liu, H. Zhang, Z. Ding, Q. Liu, C. Zhu, A comprehensive active  
735 learning method for multiclass imbalanced data streams with concept  
736 drift, Knowledge-Based Systems 215 (2021) 106778.

- [13] A. Tharwat, W. Schenck, Balancing exploration and exploitation: A novel active learner for imbalanced data, *Knowledge-Based Systems* 210 (2020) 106500.
- [14] Y.-Y. Kim, K. Song, J. Jang, I.-c. Moon, Lada: Look-ahead data acquisition via augmentation for deep active learning, *Advances in Neural Information Processing Systems* 34 (2021).
- [15] G. Douzas, F. Bacao, Geometric SMOTE a geometrically enhanced drop-in replacement for SMOTE, *Information Sciences* 501 (2019) 118–135.
- [16] J. Katz-Samuels, J. Zhang, L. Jain, K. Jamieson, Improved algorithms for agnostic pool-based active classification, in: *International Conference on Machine Learning*, PMLR, 2021, pp. 5334–5344.
- [17] T. Su, S. Zhang, T. Liu, Multi-spectral image classification based on an object-based active learning approach, *Remote Sensing* 12 (2020) 504.
- [18] J. Fonseca, G. Douzas, F. Bacao, Increasing the Effectiveness of Active Learning: Introducing Artificial Data Generation in Active Learning for Land Use/Land Cover Classification, *Remote Sensing* 2021, Vol. 13, Page 2619 13 (13) (2021) 2619.
- [19] D. Yoo, I. S. Kweon, Learning loss for active learning, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 93–102.
- [20] H. H. Aghdam, A. Gonzalez-Garcia, A. Lopez, J. Weijer, Active learning for deep detection neural networks, in: *Proceedings of the IEEE International Conference on Computer Vision*, Vol. 2019-Octob, 2019, pp. 3671–3679.
- [21] B. Gu, Z. Zhai, C. Deng, H. Huang, Efficient active learning by querying discriminative and representative samples and fully exploiting unlabeled data, *IEEE Transactions on Neural Networks and Learning Systems* 32 (2021) 4111–4122.
- [22] P. Kumar, A. Gupta, Active learning query strategies for classification, regression, and clustering: A survey, *Journal of Computer Science and Technology* 2020 35:4 35 (2020) 913–945.

- [23] Y. Fu, X. Zhu, B. Li, A survey on instance selection for active learning, Knowledge and information systems 35 (2) (2013) 249–283.
- [24] A. Samat, P. Gamba, S. Liu, P. Du, J. Abuduwaili, Jointly informative and manifold structure representative sampling based active learning for remote sensing image classification, IEEE Transactions on Geoscience and Remote Sensing 54 (2016) 6803–6817.
- [25] S. J. Huang, R. Jin, Z. H. Zhou, Active learning by querying informative and representative examples, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (2014) 1936–1949.
- [26] D. Ienco, A. Bifet, I. Žliobaite, B. Pfahringer, Clustering based active learning for evolving data streams, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8140 LNAI (2013) 79–93.
- [27] K. Brinker, Incorporating diversity in active learning with support vector machines, in: Proceedings of the 20th international conference on machine learning (ICML-03), 2003, pp. 59–66.
- [28] J. Jia, M. T. Schaub, S. Segarra, A. R. Benson, Graph-based semi-supervised & active learning for edge flows, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 761–771.
- [29] Z. del Rosario, M. Rupp, Y. Kim, E. Antono, J. Ling, Assessing the frontier: Active learning, model accuracy, and multi-objective candidate discovery and optimization, The Journal of Chemical Physics 153 (2020) 024112.
- [30] B. Settles, From theories to queries: Active learning in practice, in: Active Learning and Experimental Design workshop In conjunction with AISTATS 2010, JMLR Workshop and Conference Proceedings, 2011, pp. 1–18.
- [31] D. D. Lewis, W. A. Gale, A sequential algorithm for training text classifiers, in: SIGIR’94, Springer, 1994, pp. 3–12.

- 799 [32] T. Luo, K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen,  
800 T. Hopkins, D. Cohn, Active learning to recognize multiple types of  
801 plankton., *Journal of Machine Learning Research* 6 (4) (2005).
- 802 [33] W. Liu, J. Yang, P. Li, Y. Han, J. Zhao, H. Shi, A novel object-based  
803 supervised classification method with active learning and random forest  
804 for polsar imagery, *Remote Sensing* 10 (7) (2018) 1092.
- 805 [34] J. Li, J. M. Bioucas-Dias, A. Plaza, Spectral-spatial classification of  
806 hyperspectral data using loopy belief propagation and active learning,  
807 *IEEE Transactions on Geoscience and remote sensing* 51 (2) (2012) 844–  
808 856.
- 809 [35] N. Abe, Query learning strategies using boosting and bagging, *Proc. of*  
810 *15<sup>th</sup> Int. Conf. on Machine Learning (ICML98)* (1998) 1–9.
- 811 [36] P. Melville, R. J. Mooney, Diverse ensembles for active learning, in: *Pro-*  
812 *ceedings of the twenty-first international conference on Machine learn-*  
813 *ing*, 2004, p. 74.
- 814 [37] M. Bloodgood, Support vector machine active learning algorithms with  
815 query-by-committee versus closest-to-hyperplane selection, *Proceedings*  
816 *- 12th IEEE International Conference on Semantic Computing, ICSC*  
817 *2018 2018-January* (2018) 148–155.
- 818 [38] J. Zhou, S. Sun, Improved margin sampling for active learning, *Com-*  
819 *munications in Computer and Information Science* 483 (2014) 120–129.
- 820 [39] S. Behpour, K. M. Kitani, B. D. Ziebart, Ada: Adversarial data augmen-  
821 tation for object detection, *Proceedings - 2019 IEEE Winter Conference*  
822 *on Applications of Computer Vision, WACV 2019* (2019) 1243–1252.
- 823 [40] Z. Zhong, L. Zheng, G. Kang, S. Li, Y. Yang, Random erasing data  
824 augmentation, in: *Proceedings of the AAAI Conference on Artificial*  
825 *Intelligence*, Vol. 34, 2020, pp. 13001–13008.
- 826 [41] T. DeVries, G. W. Taylor, Dataset augmentation in feature space, in:  
827 *5th International Conference on Learning Representations, ICLR 2017*  
828 *- Workshop Track Proceedings, International Conference on Learning*  
829 *Representations, ICLR*, 2017.

- [42] C. Shorten, T. M. Khoshgoftaar, A survey on image data augmentation for deep learning, *Journal of Big Data* 6 (1) (2019) 1–48.
- [43] B. K. Iwana, S. Uchida, An empirical survey of data augmentation for time series classification with neural networks, *Plos one* 16 (7) (2021) e0254841.
- [44] S. C. Wong, A. Gatt, V. Stamatescu, M. D. McDonnell, Understanding data augmentation for classification: when to warp?, in: *2016 international conference on digital image computing: techniques and applications (DICTA)*, IEEE, 2016, pp. 1–6.
- [45] O. Kashefi, R. Hwa, Quantifying the evaluation of heuristic methods for textual data augmentation, in: *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, Association for Computational Linguistics, Online, 2020, pp. 200–208.
- [46] 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.
- [47] J. Fonseca, G. Douzas, F. Bacao, Improving imbalanced land cover classification with k-means smote: Detecting and oversampling distinctive minority spectral signatures, *Information* 12 (7) (2021) 266.
- [48] G. Douzas, F. Bacao, J. Fonseca, M. Khudinyan, Imbalanced learning in land cover classification: Improving minority classes’ prediction accuracy using the geometric smote algorithm, *Remote Sensing* 11 (24) (2019) 3040.
- [49] H. Han, W.-Y. Wang, B.-H. Mao, Borderline-smote: A new over-sampling method in imbalanced data sets learning, in: *International Conference on Intelligent Computing*, Springer, Berlin, Heidelberg, 2005, pp. 878–887.
- [50] G. Douzas, F. Bacao, F. Last, Improving imbalanced learning through a heuristic oversampling method based on k-means and smote, *Information Sciences* 465 (2018) 1–20.



- [51] Y. Ma, S. Lu, E. Xu, T. Yu, L. Zhou, Combining active learning and data augmentation for image classification, in: Proceedings of the 2020 3rd International Conference on Big Data Technologies, 2020, pp. 58–62.
- [52] H. Quteineh, S. Samothrakis, R. Sutcliffe, Textual data augmentation for efficient active learning on tiny datasets, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 7400–7410.
- [53] Q. Li, Z. Huang, Y. Dou, Z. Zhang, A framework of data augmentation while active learning for chinese named entity recognition, in: International Conference on Knowledge Science, Engineering and Management, Springer, 2021, pp. 88–100.
- [54] C. Wu, The decision tree approach to classification., Purdue University, 1975.
- [55] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.
- [56] T. K. Ho, Random decision forests, in: Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1, ICDAR '95, IEEE Computer Society, USA, 1995, p. 278.
- [57] J. A. Nelder, R. W. Wedderburn, Generalized linear models, Journal of the Royal Statistical Society: Series A (General) 135 (3) (1972) 370–384.
- [58] L. A. Jeni, J. F. Cohn, F. De La Torre, Facing imbalanced data - Recommendations for the use of performance metrics, in: Proceedings - 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction, ACII 2013, 2013, pp. 245–251.
- [59] M. Fatourechi, R. K. Ward, S. G. Mason, J. Huggins, A. Schloegl, G. E. Birch, Comparison of evaluation metrics in classification applications with imbalanced datasets, in: 2008 seventh international conference on machine learning and applications, IEEE, 2008, pp. 777–782.
- [60] M. Kubat, S. Matwin, et al., Addressing the curse of imbalanced training sets: one-sided selection, in: Icml, Vol. 97, Citeseer, 1997, pp. 179–186.

- 890 [61] D. Kottke, A. Calma, D. Huseljic, G. Kreml, B. Sick, Challenges of  
891 reliable, realistic and comparable active learning evaluation, in: CEUR  
892 Workshop Proceedings, Vol. 1924, 2017, pp. 2–14.
- 893 [62] T. Reitmaier, B. Sick, Let us know your decision: Pool-based active  
894 training of a generative classifier with the selection strategy 4ds, Infor-  
895 mation Sciences 230 (2013) 106–131.
- 896 [63] J.-F. Kagy, T. Kayadelen, J. Ma, A. Rostamizadeh, J. Strnadova, The  
897 practical challenges of active learning: Lessons learned from live exper-  
898 imentation, arXiv preprint arXiv:1907.00038 (6 2019).
- 899 [64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion,  
900 O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vander-  
901 plas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay,  
902 Scikit-learn: Machine Learning in Python, Journal of Machine Learning  
903 Research 12 (Oct) (2011) 2825–2830.
- 904 [65] G. Lemaître, F. Nogueira, C. K. Aridas, Imbalanced-learn: A python  
905 toolbox to tackle the curse of imbalanced datasets in machine learning,  
906 Journal of Machine Learning Research 18 (17) (2017) 1–5.
- 907 [66] J. Demšar, Statistical comparisons of classifiers over multiple data sets,  
908 Journal of Machine Learning Research 7 (2006) 1–30.
- 909 [67] M. Friedman, The use of ranks to avoid the assumption of normality  
910 implicit in the analysis of variance, Journal of the american statistical  
911 association 32 (200) (1937) 675–701.
- 912 [68] F. Wilcoxon, Individual Comparisons by Ranking Methods, Biometrics  
913 Bulletin 1 (6) (1945) 80.
- 914 [69] S. Holm, A simple sequentially rejective multiple test procedure, Scan-  
915 dinavian journal of statistics (1979) 65–70.