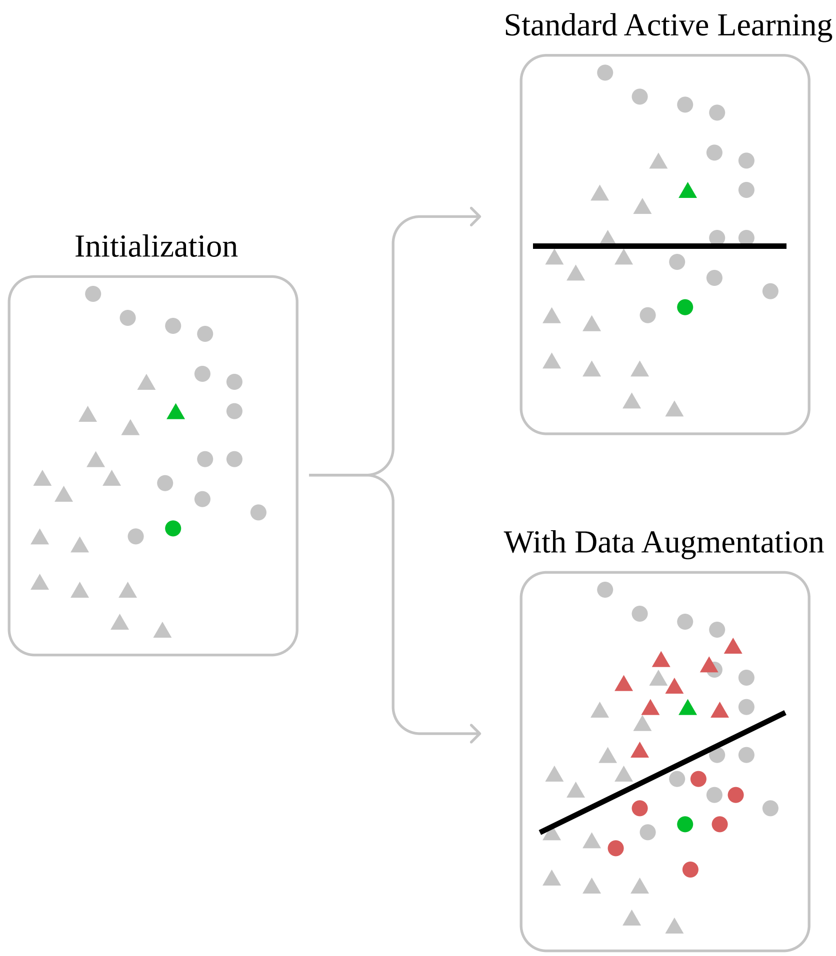
Graphical Abstract

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[a], Fernando Bacao[a]

[a]*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 the 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 the 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 2.3) 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 2.2). We also propose a hyperparameter optimization module, which 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 introduces relevant topics discussed in the paper and describes the related work. Section 3 describes the proposed method. Section 4 describes the methodology of the study's experiment. Section 5 presents the results obtained from the experiment, as well as a discussion of these results. Section 6 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 $\mathcal{D}^t_{lab}$ to produce, for each $x_i \in \mathcal{D}^t_{pool}$, 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 ucertainty from $\mathcal{D}^t_{pool}$ to form $\mathcal{D}^t_{new}$. The iteration ends with the update of $\mathcal{D}^{t+1}_{lab} = \mathcal{D}^t_{lab} \cup \mathcal{D}^t_{new}$ and $\mathcal{D}^{t+1}_{pool} = \mathcal{D}^t_{pool} \setminus \mathcal{D}^t_{new}$ [17, 2]. This process is shown in Figure 1. Before the start of the iterative process, assuming $\mathcal{D}^{t=0}_{lab} = \emptyset$, the data used to populate $\mathcal{D}^{t=1}_{lab}$ is typically collected randomly from $\mathcal{D} = \mathcal{D}^{t=0}_{pool}$ and is labeled by a supervisor [18, 19, 20].

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

1. Informative-based. These strategies use the classifier's output to assess the importance of each observation towards the performance of the classifier [23].

2. Representative-based. These strategies estimate the the optimal set of observations that will optimize the classifier's performance [22].

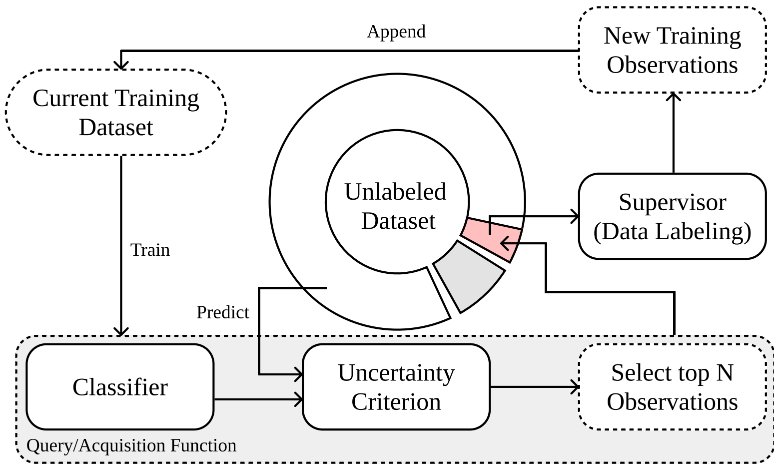


Figure 1: Diagram depicting a typical AL iteration. In the first iteration, the training set collected during the initialization process becomes the “Current Training Dataset”.

Although there are significant contributions toward the development of more robust query functions and classifiers in AL, modifications to AL’s basic structure are rarely explored. In [19] the authors introduce a loss prediction module in the AL framework to replace the uncertainty criterion. This model implements a second classifier to predict the expected loss of the unlabeled observations (using the actual losses collected during the training of the original classifier) and return the unlabeled observations with the highest expected loss. However, this contribution is specific to neural networks (and more specifically, to deep neural networks)and was only tested for image classification.

2.2. Data Augmentation

Data Augmentation methods expand the training dataset by introducing new and informative observations [24]. The production of artificial data may be done via the introduction of perturbations on the input [25], feature [26], or output space [24]. Data Augmentation methods may be divided into two categories [27]:

1. Heuristic approaches attempt to generate new and relevant observations through the application of a predefined procedure, usually incor-

porating some degree of randomness [28]. Since these methods typically occur in the input space, they require less data and computational power when compared to Neural Network methods.

2. Neural Network approaches, on the other hand, map the original input space into a lower-dimensional representation, known as the feature space [26]. 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.

While some techniques may depend on the domain, others are domain-agnostic. For example, Random Erasing [25], Translation, Cropping and Flipping are examples of image data-specific augmentation methods. Other methods, such as autoencoders, may be considered domain agnostic.

2.3. Data Augmentation in Active Learning

The standard AL model can be complemented with a data augmentation function, $f_{aug}(x_i;\tau)$, where τ defines the augmentation policy. In this context, τ refers to the transformation applied and its hyperparameters and $f_{aug}(x;\tau) : \mathcal{D} \to \mathcal{D}_{aug}(\mathcal{D})$ produces a modified observation, $\tilde{x} \in \mathcal{D}_{aug}(\mathcal{D})$ where $\mathcal{D}_{aug}(\mathcal{D})$ is the set of modified observations. This involves the usage of a new set of data, $\mathcal{D}^t_{train} = \mathcal{D}^t_{lab} \cup \mathcal{D}^t_{aug}$, to train the classifier.

As found in Section 2.1, improvements proposed in the AL framework are mostly focused on modifications of the classifier or query strategy. Furthermore, the few recent AL contributions implementing data augmentation were all (except one) applied to the computer vision or natural language processing (NLP) realm.

The only AL model found that uses data augmentation outside of the computer vision or NLP domains uses a pipelined approach, described in [18]. In this study, the AL model proposed is applied for tabular data using an oversampling data augmentation policy (*i.e.*, the artificial data was generated only to balance the target class frequencies). However, this AL model was applied in a Land Use/Land Cover context with 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, this method does not allow augmentation policy optimization (i.e., every hyperparameter has to be hardcoded a priori).

The Bayesian Generative Active Deep Learning (BGDAL) [29] is another example of a pipelined combination of f_{acq} and f_{aug}, applied image classification. BGDAL uses a Variational AutoEncoder (VAE) architecture to generate artificial observations. However, the proposed model is computationally expensive, requires a large data pool to train the VAE, and is not only dependent on the quality of the augmentations performed, but also on the performance of the discriminator and classifiers used.

The method proposed in [14], Look-Ahead Data Acquisition for Deep Active Learning, implement data augmentation to train a deep-learning classifier. However, adapting existing AL applications to use this approach is often impractical and implies the usage of image data, since the augmentations used are image data specific and occur on the unlabeled observations, before the unlabeled data selection.

The Variational Adversarial Active Learning (VAAL) model [30] is a deep AL approach to image classification that uses as inputs the embeddings produced by a VAE into a secondary classifier, working as f_{acq}, to predict if $x_i \in \mathcal{D}$ belongs to $\mathcal{D}_{pool}$. The n true positives with the highest certainty are labeled by the supervisor and $\mathcal{D}_{pool}$ and $\mathcal{D}_{lab}$ are updated as described in Section 2.1. The Task-aware VAAL model [31] extends the VAAL model by introducing a ranker, which consists of the Learning Loss module introduced in [19]. These models use data augmentation techniques to train the different neural network-based components of the proposed models. However, the AL components used are specific image classification, computationally expensive and the analysis of the effect of data augmentation in these AL models is not discussed.

In [32] the proposed AL method was designed specifically for image data classification, where a deep learning model was implemented as a classifier, but its architecture is not described, the augmentation policies used are unknown and the results reported correspond to single runs of the discussed model. The remaining AL models found implementing data augmentation were introduced for NLP applications, in [33, 34]. However, these methods were designed for specific applications within that domain and are not necessarily transferable to other domains or tasks.

3. 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. In addition, there is also a significant variability of the data selection efficiency during different runs of the AL iterative process [18].

This paper provides a context-agnostic AL framework for 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.

4. Analysis of the impact of Data Augmentation and Oversampling in AL over 10 different datasets of different domains, while comparing them with the standard AL framework.

The proposed AL framework is depicted in Figure 2. The generator element becomes an additional source of data and is expected to introduce additional data variability into the training dataset. This should allow the classifier to generalize better and perform more consistently over unseen observations. However, in this scenario, the amount of data to generate per class at each iteration is unknown. Consequently, the hyperparameter tuning step was introduced to estimate the optimal data augmentation policy at each iteration. In our implementation, this step uses the current training dataset to perform an exhaustive search over specified parameters of the generator, tested over a 5-fold cross-validation method. The best augmentation policy found is used to train the iteration's classifier in the following step. This procedure is described in Algorithm 1.

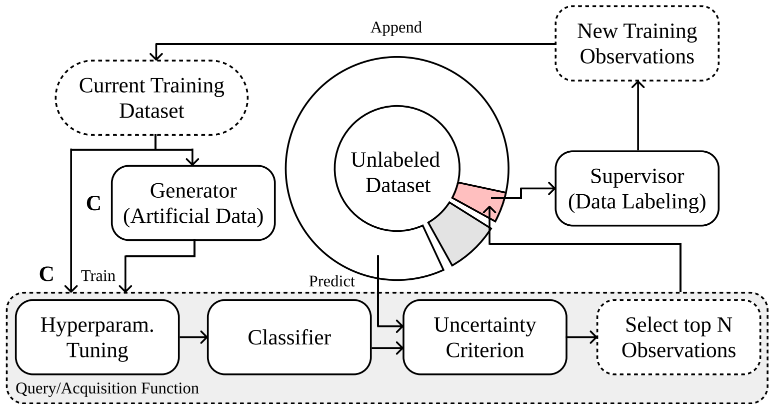


Figure 2: Diagram depicting the proposed AL iteration. The proposed modifications are marked with a boldface "C".

To show the effectiveness of data augmentation in an AL implementation, we implemented a simple modification in the selection mechanism of the G-SMOTE algorithm. We use the uncertainties produced by f_{acq} to compute the probabilities of observations to be selected for augmentation, as an additional parameter. This modification is described in Algorithm 2

This modification facilitates the usage of G-SMOTE beyond its original oversampling purposes. However, in this paper, the data augmentation strategies are also used to ensure that class frequencies are balanced. Furthermore, the amount of artificial data produced for each class is defined by the *augmentation factor*, α_{af}, which represents a percentage of the majority class C_{maj} (*e.g.,* an augmentation factor of 1.2 will ensure there are $count(C_{maj}) \times 1.2$ observations in every class). In this paper's experiment, the data generation mechanism is similar to the one in [18]. This allows the direct comparison of the two frameworks and establishes a causality of the performance variations to the data generation mechanism (*i.e.,* augmentation vs normal oversampling) and hyperparameter tuning steps. However, in this case, the hyperparameter tuning is only going to be used for augmentation policy optimization.

Algorithm 1: Proposed AL Framework (Single iteration)

Given: $t \geq 1$, performance metric f_{pm}
Input: $\mathcal{D}_{pool}$, $\mathcal{D}_{lab}$, f_c, f_{aug}, f_{acq}, τ_{grid}, k, n
Output: $\mathcal{D}_{pool}$, $\mathcal{D}_{lab}$

Function *ParameterTuning*(f_c, f_{aug}, τ_{grid}, $\mathcal{D}_{lab}$, k)**:**
  $p \leftarrow 0$
  $\tau \leftarrow \emptyset$
  $\{\mathcal{D}_{lab}^1, \ldots \mathcal{D}_{lab}^k\} \leftarrow \mathcal{D}_{lab}$ // $\mathcal{D}_{lab}^n \cap \mathcal{D}_{lab}^m = \emptyset, \forall (n,m) \in 1, \ldots, k$
  forall $\tau' \in \tau_{grid}$ **do**
    $p' \leftarrow \emptyset$
    forall $\mathcal{D}_{lab}^i \in \{\mathcal{D}_{lab}^1, \ldots \mathcal{D}_{lab}^k\}$ **do**
      $\mathcal{D}'_{test} \leftarrow \mathcal{D}_{lab}^i$
      $\mathcal{D}'_{train} \leftarrow \mathcal{D}_{lab} \setminus \mathcal{D}_{lab}^i$
      $\mathcal{D}'_{train} \leftarrow f_{aug}(\mathcal{D}'_{train}; \tau')$
      train f_c using $\mathcal{D}'_{train}$
      $p' \leftarrow p' \cup \{f_{pm}(f_c(\mathcal{D}_{test}))\}$
    $p' \leftarrow \frac{\sum_{x_i \in p'} x_i}{k}$
    if $p' > p$ **then**
      $p \leftarrow p'$
      $\tau \leftarrow \tau'$
  return τ

begin
  $\tau \leftarrow ParameterTuning(f_c, f_{aug}, \tau_{grid}, \mathcal{D}_{lab}, k)$
  $\mathcal{D}_{train} \leftarrow f_{aug}(\mathcal{D}_{lab}; \tau)$
  train f_c using $\mathcal{D}_{train}$
  $\mathcal{D}_{new} = \arg\max_{\mathcal{D}'_{pool} \subset \mathcal{D}_{pool}, |\mathcal{D}'_{pool}|=n} \sum_{x \in \mathcal{D}'_{pool}} f_{acq}(x; f_c)$
  annotate $\mathcal{D}_{new}$
  $\mathcal{D}_{pool} \leftarrow \mathcal{D}_{pool} \setminus \mathcal{D}_{new}$
  $\mathcal{D}_{lab} \leftarrow \mathcal{D}_{lab} \cup \mathcal{D}_{new}$

Algorithm 2: G-SMOTE Modified for Data Augmentation in AL

Given: $t \geq 1$, $\mathcal{D}^t_{lab} \neq \emptyset$, $\mathcal{D}_{lab} = \mathcal{D}^{min}_{lab} \cup \mathcal{D}^{maj}_{lab}$, $GSMOTE$

Input: $\mathcal{D}^t_{pool}$, $\mathcal{D}^t_{lab}$, f_c^{t-1}, f_{acq}, τ

Output: $\mathcal{D}^t_{train}$

Function *DataSelection*($\mathcal{D}^t_{lab}$, f_{acq}, f_c^{t-1})**:**

  $U \leftarrow \emptyset$

  $P \leftarrow \emptyset$

  $p_s \sim \mathcal{U}(0, 1)$

  forall $x_i \in \mathcal{D}^t_{lab}$ **do**

    $u_{x_i} \leftarrow f_{acq}(x_i; f_c^{t-1})$

    $U \leftarrow U \cup \{u_{x_i}\}$

  forall $u_{x_i} \in U$ **do**

    $p_{x_i} \leftarrow \frac{u_{x_i}}{\sum U} + \sum P$

    $P \leftarrow P \cup \{p_{x_i}\}$

  $i \leftarrow argmax(P < p_s)$

  return i-th element in $\mathcal{D}^t_{lab}$

begin

  $\mathcal{D}^{min}_{aug} \leftarrow \emptyset$

  $\mathcal{D}^{maj}_{aug} \leftarrow \emptyset$

  $\alpha_{af}, \alpha_{trunc}, \alpha_{def} \leftarrow \tau$

  $N \leftarrow count(C_{maj}) \times \alpha_{af}$

  forall $\mathcal{D}'_{aug} \in \{\mathcal{D}^{min}_{aug}, \mathcal{D}^{maj}_{aug}\}$, $\mathcal{D}'_{lab} \in \{\mathcal{D}^{min}_{lab}, \mathcal{D}^{maj}_{lab}\}$ **do**

    while $|\mathcal{D}'_{aug}| < N$ **do**

      $x_{center} \leftarrow DataSelection(\mathcal{D}'_{lab}, f_{acq}, f_c^{t-1})$

      $x_{gen} \leftarrow GSMOTE(x_{center}, \mathcal{D}^t_{lab}, \alpha_{trunc}, \alpha_{def})$

      $\mathcal{D}'_{aug} \leftarrow \mathcal{D}'_{aug} \cup \{x_{gen}\}$

  $\mathcal{D}_{aug} \leftarrow \mathcal{D}^{min}_{aug} \cup \mathcal{D}^{maj}_{aug}$

  $\mathcal{D}^t_{train} \leftarrow \mathcal{D}^t_{lab} \cup \mathcal{D}_{aug}$

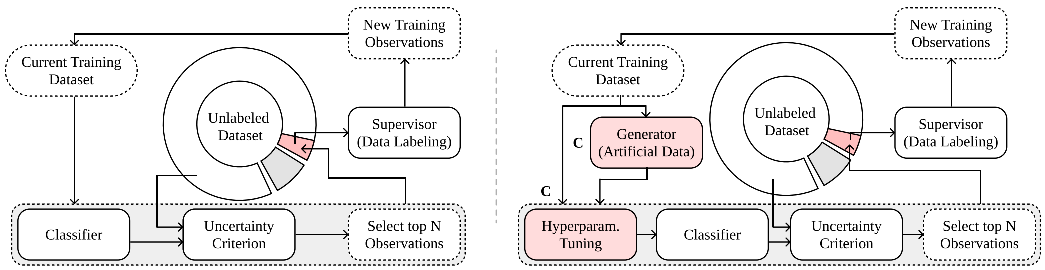


Figure 3: Simplified diagrams highlighting the differences between the proposed and standard AL iterations. The proposed modifications are highlighted in red and marked with a boldface "C".

The comparison of diagrams between the proposed and standard AL frameworks is shown in Figure 3. In the proposed framework, we (1) generalize the generator module to accept any data augmentation method or policy and (2) a hyperparameter tuning module to estimate the optimal data augmentation policy. This framework was designed to be task-agnostic. Specifically, any data augmentation method (domain-specific or not) may be used, as well as any other parameter search method. It is also expected to be compatible with other AL modifications, including the ones that do not affect solely the classifier or uncertainty criterion, such as the one proposed in [19].

4. Methodology

This section describes the different elements included in the experimental procedure. The datasets used were acquired in open data repositories. Their sources and preprocessing steps are defined in Subsection 4.1. The classifiers used in the experiment are defined in Subsection 4.2. The metrics chosen to measure AL performance and overall classification performance are defined in Subsection 4.3. The experimental procedure is described in Subsection 4.4. The implementation of the experiment and resources used to do so are described in Subsection 4.5.

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

4.1. Datasets

The datasets used to test the proposed method are publicly available in open data repositories. Specifically, they were retrieved from OpenML and the UCI Machine Learning Repository. They were chosen considering different domains of application, imbalance ratios, dimensionality and number of target classes, all of them focused on classification tasks. The goal is to demonstrate the performance of the different AL frameworks in various scenarios and domains. The data preprocessing approach was similar across all datasets. Table 1 describes the key properties of the 10 preprocessed datasets 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

The data preprocessing pipeline is depicted as a flowchart in Figure 4. The missing values are removed from each dataset by removing the corresponding observations. This ensures that the input data in the experiment is kept as close to its original form as possible. The non-metric features (*i.e.,* binary, categorical, and ordinal variables) were removed since the application of G-SMOTE is limited to continuous and discrete features. The datasets containing over 2000 observations were downsampled in order to maintain the datasets to a manageable size. The data sampling procedure preserves the relative class frequency of the dataset, in order to maintain the Imbalance Ratio (IR) originally found in each dataset (where $IR = \frac{count(C_{maj})}{count(C_{\min})}$). The remaining features of each dataset are scaled to the range of $[-1, 1]$ to ensure a common range across features.

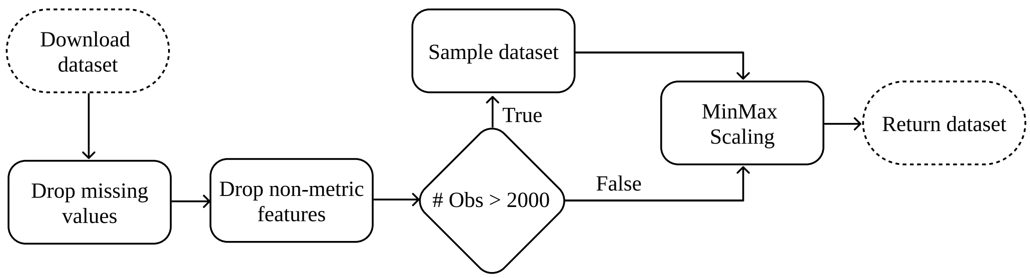


Figure 4: Data preprocessing pipeline.

The preprocessed datasets were stored into a SQLite database file and is available along with the experiment's source code in the GitHub repository of the project (see Subsection 4.5).

4.2. Machine Learning Algorithms

We used a total of 4 classification algorithms and a heuristic data augmentation mechanism. The choice of classifiers was based on the popularity and family of the classifiers (tree-based, nearest neighbors-based, ensemble-based and linear models). Our proposed method was tested using a Decision Tree (DT) [35], a K-nearest neighbors classifier (KNN) [36], a Random Forest Classifier (RF) [37] and a Logistic Regression (LR) [38]. Since the target variables are multi-class, the LR classifier was implemented using the one-versus-all approach. The predicted class is assigned to the label with the 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 3.

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 3.

4.3. 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 to quantify a model's classification performance [39]. 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 [40]. However, these metrics remain popular choices for the evaluation of classification performance. Other performance metrics like $Precision = \frac{TP}{TP+TN}$, $Recall = \frac{TP}{TP+FN}$ 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 [39, 41], 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 [41]. Both metrics are calculated in a multiclass context considering a one-versus-all approach. For multiclass problems, the G-mean scorer is calculated as its average per class values:

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

- The F-score metric consists of the harmonic mean of Precision and Recall. The two metrics are also calculated considering a one-versus-all approach. The F-score for the multi-class case can be calculated using its average per class values [39]:

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

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

$$OA = \frac{\sum_{c} \mathrm{TP}_c}{\sum_{c} (\mathrm{TP}_c + \mathrm{FP}_c)}$$

The comparison of the performance of AL frameworks is based on its data selection and augmentation efficacy. Specifically, an efficient data selection/generation policy allows the production of classifiers with high performance on unseen data while using as least non-artificial training data as possible. To measure the performance of the different AL setups, we follow the recommendations found in [42]. The performance of an AL setup will be compared using two AL-specific performance metrics:

- Area Under the Learning Curve (AULC). It is the sum of the classification performance over a validation/test set of the classifiers trained of all AL iterations. To facilitate the interpretability of this metric, the resulting AULC scores are fixed within the range $[0, 1]$ by dividing the AULC scores by the total amount of iterations (*i.e.*, the maximum performance area).

- Data Utilization Rate (DUR) [43]. Measures the percentage of training 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.

4.4. 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 [44]. In this scenario, since the dataset is already labeled, the annotation process is done at zero cost. Figure 5 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 5 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).

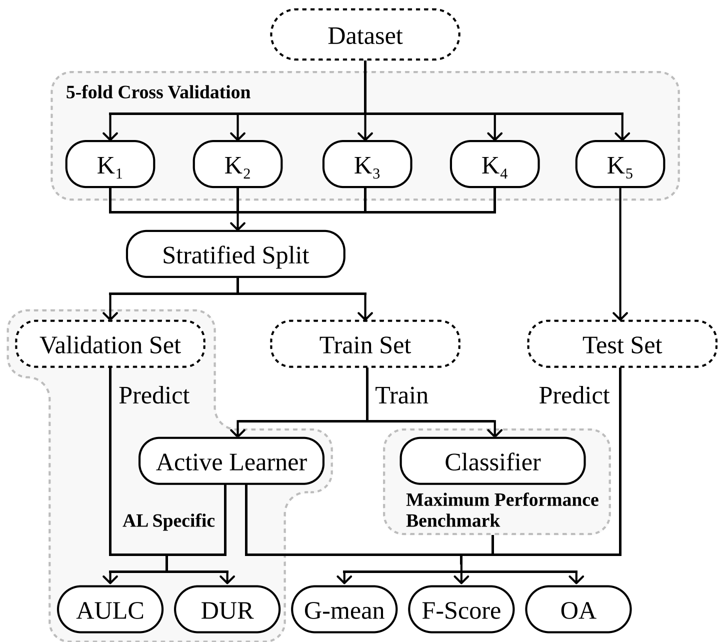


Figure 5: Experimental procedure flowchart. The preprocessed datasets are split into five folds. One of the folds is used to test the best-found classifiers using AL and the classifiers trained using the entire training dataset (containing the remaining folds). The training set is used to run both the 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.

The hyperparameters defined for the AL frameworks, Classifiers, and Generators are shown in Table 2. In the Generators table, we distinguish the G-SMOTE algorithm working as a normal oversampling method from G-SMOTE-AUGM, which performs generates additional artificial data on top of the usual oversampling mechanism. Since the G-SMOTE-AUGM method is intended to be used with varying parameter values (via within-iteration parameter tuning), the parameters were defined as a list of various possible 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
	penalty	L2 (Ridge)
KNN	# 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.

4.5. Software Implementation

The experiment was implemented using the Python programming language, along with the Python libraries Scikit-Learn [45], Imbalanced-Learn [46], Geometric-SMOTE [15], Research-Learn and ML-Research libraries. All functions, algorithms, experiments, and results are provided in the GitHub repository of the project.

5. Results & Discussion

In a multiple dataset experiment, the analysis of results should not rely uniquely on the average performance scores across datasets. The domain of application and fluctuations of performance scores between datasets make the analysis of these averaged results less accurate. Instead, it is generally recommended to use the mean ranking scores to extend the analysis [47]. Since mean performance scores are still intuitive to interpret, we will present and discuss both results. The rank values are assigned based on the mean scores of 3 different runs of 5-fold Cross-Validation (15 performance estimations per dataset) for each combination of dataset, AL configuration, classifier, and performance metric.

5.1. Results

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

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

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.

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

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

The DUR scores relative to the Standard AL method are shown in Figure 6. A DUR below 1 means that the Proposed/Oversampling method requires less data than the Standard AL method to reach the same performance threshold. For example, running an AL simulation using the KNN classifier requires 69.6% of the amount of data required by the Standard AL method using the same classifier to reach an F-Score of 0.62 (*i.e.,* requires 30.4% less data).

The mean optimal classification scores of AL methods and Classifiers

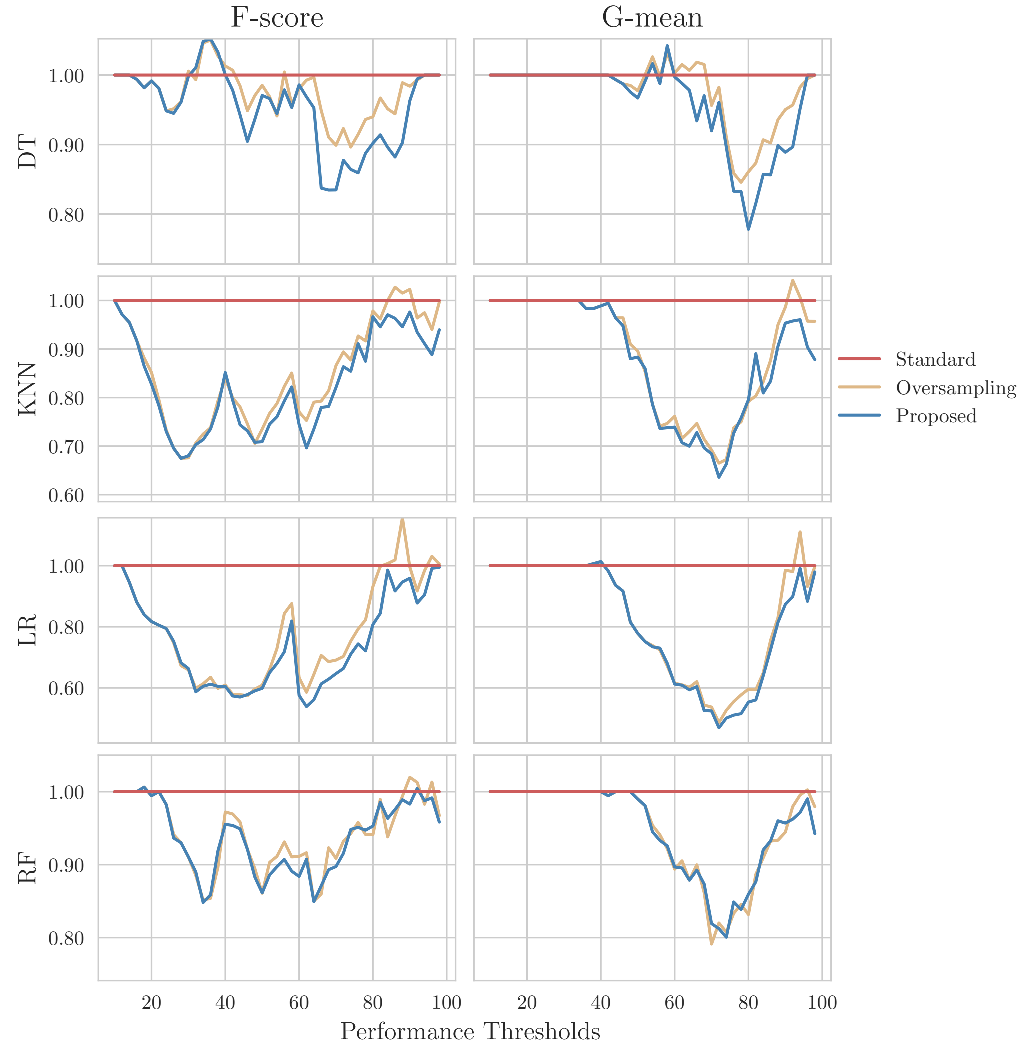


Figure 6: Mean data utilization rates. The y-axis shows the percentage of data (relative to the baseline AL framework) required to reach the different performance thresholds.

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

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).

5.2. 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 [47]. Overall, we perform 3 statistical tests. The Friedman test [48] 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 [49] 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 [50] 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%	**3.1%**	3.2%
0.60	KNN	3.6%	2.6%	**2.5%**
0.60	LR	3.9%	**2.2%**	**2.2%**
0.60	RF	2.4%	**2.1%**	**2.1%**
0.66	DT	4.6%	4.6%	**4.2%**
0.66	KNN	4.9%	3.7%	**3.5%**
0.66	LR	5.7%	3.2%	**3.1%**
0.66	RF	3.0%	2.8%	**2.7%**
0.70	DT	6.6%	6.1%	**5.8%**
0.70	KNN	8.5%	5.0%	**4.7%**
0.70	LR	9.5%	4.6%	**4.3%**
0.70	RF	4.5%	**3.2%**	3.3%
0.76	DT	16.5%	13.0%	**12.7%**
0.76	KNN	17.8%	9.7%	**9.0%**
0.76	LR	16.6%	10.0%	**7.8%**
0.76	RF	10.1%	**5.5%**	**5.5%**
0.80	DT	36.1%	30.4%	**27.1%**
0.80	KNN	22.7%	18.0%	**17.8%**
0.80	LR	25.2%	16.0%	**14.2%**
0.80	RF	15.5%	**9.0%**	9.5%
0.86	DT	60.5%	56.7%	**54.5%**
0.86	KNN	39.9%	**37.0%**	37.8%
0.86	LR	32.6%	27.5%	**27.0%**
0.86	RF	28.0%	**25.7%**	**25.7%**
0.90	DT	72.5%	70.7%	**67.8%**
0.90	KNN	49.9%	50.3%	**49.3%**
0.90	LR	52.5%	53.8%	**49.3%**
0.90	RF	44.6%	**42.6%**	43.5%
0.96	DT	100.0%	**99.5%**	100.0%
0.96	KNN	79.4%	75.6%	**71.6%**
0.96	LR	87.5%	83.1%	**79.8%**
0.96	RF	63.6%	64.2%	**63.1%**

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

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

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

5.3. 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

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

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

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).

performance differences between the proposed method and both the standard and oversampling methods. The Friedman test in Table 7 showed that the difference in the performance of these AL frameworks are statistically significant, regardless of the classifier or performance metric being used.

The proposed method showed more consistent data utilization requirements to most of the assessed G-mean score thresholds when compared to the remaining AL methods, as seen in Table 5. For example, to reach a G-mean Score of 0.9 using the KNN and LR classifiers, the average amount of data required with the Oversampling AL approach increased when compared to the Standard approach. However, the proposed method was able to decrease the amount of data required in both situations. The robustness of the Proposed method is clearer in Figure 6. In most cases, this method was able to outperform the Oversampling method. At the same time, the proposed method also addresses inconsistencies in situations where the Oversampling method was unable to outperform the standard method.

The statistical analyses found in Tables 8 and 9 showed that the proposed method's superiority was statistically significant in all datasets except one (Baseball) and established statistical significance when compared to the Standard AL method for all combinations of classifier and performance metric, including when the Oversampling AL method failed to do so. These results show that the Proposed method increased the reliability of the new AL framework and improved the quality of the final classifier while using less data.

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

This study discussed the effect of data augmentation within the AL framework, along with the exploration of optimal augmentation methods within AL iterations. However, the conceptual nature of this study implies some limitations. Specifically, the large number of experiments required to test the method's efficacy, along with the limited computational power available, led to a limited exploration of the grid search's potential. Future work should focus on 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 at 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 in 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 5.1), which is consistent with the findings in [18, 42]. 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.

6. 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 of data augmentation are domain and/or task-specific. These methods typically apply deep learning approaches to 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 a performance as good as (or better than) classifiers trained with the entire training dataset (*i.e.,* without using AL). In addition, the proposed method reduced the size of the training dataset, which is expanded with artificial data.

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

Declarations

Funding

This research was supported by three research grants of the Portuguese Foundation for Science and Technology ("Fundação para a Ciência e a Tecnologia"), references SFRH/BD/151473/2021, DSAIPA/DS/0116/2019 and PCIF/SSI/0102/2017.

Code availability

The analyses and source code is available at github.com/joaopfonseca/ml-research.

References

[1] V. Nath, D. Yang, B. A. Landman, D. Xu, H. R. Roth, Diminishing uncertainty within the training pool: Active learning for medical image segmentation, IEEE Transactions on Medical Imaging 40 (10) (2021) 2534–2547.

[2] Y. Sverchkov, M. Craven, A review of active learning approaches to experimental design for uncovering biological networks, PLoS Computational Biology 13 (2017) e1005466.

[3] X. Li, D. Kuang, C. X. Ling, Active learning for hierarchical text classification, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7301 LNAI (2012) 14–25.

[4] Y. Li, J. Yin, L. Chen, Seal: Semisupervised adversarial active learning on attributed graphs, IEEE Transactions on Neural Networks and Learning Systems 32 (7) (2021) 3136–3147.

[5] O. Siméoni, M. Budnik, Y. Avrithis, G. Gravier, Rethinking deep active learning: Using unlabeled data at model training, Proceedings - International Conference on Pattern Recognition (2020) 1220–1227.

[6] J. E. Van Engelen, H. H. Hoos, A survey on semi-supervised learning, Machine Learning 109 (2) (2020) 373–440.

[7] O. Sener, S. Savarese, Active learning for convolutional neural networks: A core-set approach, in: International Conference on Learning Representations, 2018.

[8] Y. Leng, X. Xu, G. Qi, Combining active learning and semi-supervised learning to construct svm classifier, Knowledge-Based Systems 44 (2013) 121–131.

[9] H. Yu, X. Yang, S. Zheng, C. Sun, Active learning from imbalanced data: A solution of online weighted extreme learning machine, IEEE Transactions on Neural Networks and Learning Systems 30 (2019) 1088–1103.

[10] W. Zong, G.-B. Huang, Y. Chen, Weighted extreme learning machine for imbalance learning, Neurocomputing 101 (2013) 229–242.

[11] J. Qin, C. Wang, Q. Zou, Y. Sun, B. Chen, Active learning with extreme learning machine for online imbalanced multiclass classification, Knowledge-Based Systems 231 (2021) 107385.

[12] W. Liu, H. Zhang, Z. Ding, Q. Liu, C. Zhu, A comprehensive active learning method for multiclass imbalanced data streams with concept 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] S. Behpour, K. M. Kitani, B. D. Ziebart, Ada: Adversarial data augmentation for object detection, Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019 (2019) 1243–1252.

[25] Z. Zhong, L. Zheng, G. Kang, S. Li, Y. Yang, Random erasing data augmentation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 13001–13008.

[26] T. DeVries, G. W. Taylor, Dataset augmentation in feature space, in: 5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings, International Conference on Learning Representations, ICLR, 2017.

[27] C. Shorten, T. M. Khoshgoftaar, A survey on image data augmentation for deep learning, Journal of Big Data 6 (1) (2019) 1–48.

[28] 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.

[29] T. Tran, T.-T. Do, I. Reid, G. Carneiro, Bayesian generative active deep learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 6295–6304.

[30] S. Sinha, S. Ebrahimi, T. Darrell, Variational adversarial active learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 5972–5981.

[31] K. Kim, D. Park, K. I. Kim, S. Y. Chun, Task-aware variational adversarial active learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 8166–8175.

[32] 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.

[33] 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.

[34] 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.

[35] C. Wu, The decision tree approach to classification., Purdue University, 1975.

[36] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.

[37] 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.

[38] J. A. Nelder, R. W. Wedderburn, Generalized linear models, Journal of the Royal Statistical Society: Series A (General) 135 (3) (1972) 370–384.

[39] 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.

[40] 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.

[41] 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.

[42] D. Kottke, A. Calma, D. Huseljic, G. Krempl, B. Sick, Challenges of reliable, realistic and comparable active learning evaluation, in: CEUR Workshop Proceedings, Vol. 1924, 2017, pp. 2–14.

[43] T. Reitmaier, B. Sick, Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4ds, Information Sciences 230 (2013) 106–131.

[44] J.-F. Kagy, T. Kayadelen, J. Ma, A. Rostamizadeh, J. Strnadova, The practical challenges of active learning: Lessons learned from live experimentation, arXiv preprint arXiv:1907.00038 (6 2019).

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

[46] G. Lemaître, F. Nogueira, C. K. Aridas, Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning, Journal of Machine Learning Research 18 (17) (2017) 1–5.

[47] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[48] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the american statistical association 32 (200) (1937) 675–701.

[49] F. Wilcoxon, Individual Comparisons by Ranking Methods, Biometrics Bulletin 1 (6) (1945) 80.

[50] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian journal of statistics (1979) 65–70.