Class Balancing for Efficient Active Learning in Imbalanced Datasets

Yaron Fairstein
Amazon / yyfairstein@gmail.com
Oren Kalinsky
Amazon / orenkalinsky@gmail.com
Zohar Karnin
Amazon / zkarnin@amazon.com
Guy Kushilevitz
Amazon / guyk@amazon.com
Alexander Libov
Amazon / alibov@amazon.com
Sofia Tolmach
Amazon / sofiato@amazon.com

Abstract

Recent developments in active learning algorithms for NLP tasks show promising results in terms of reducing labelling complexity. In this paper we extend this effort to imbalanced datasets; we bridge between the active learning approach of obtaining diverse and informative examples, and the heuristic of class balancing used in imbalanced datasets. We develop a novel tune-free weighting technique that can be applied to various existing active learning algorithms, adding a component of class balancing. We compare several active learning algorithms to their modified version on multiple public datasets and show that when the classes are imbalanced, with manual annotation effort remaining equal the modified version significantly outperforms the original both in terms of the test metric and the number of obtained minority examples. Moreover, when the imbalance is mild or non-existent (classes are completely balanced), our technique does not harm the base algorithms.

1 Introduction

Pre-trained Language Models (PLMs) and Masked Language Models (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019) have revolutionized NLP by supplying meaningful contextual embeddings for tokens and sentences. These models pushed SoTA on many tasks, proving especially effective for text classification tasks (Sun et al., 2019). For many of these tasks, given enough labeled data for fine-tuning a PLM, satisfactory results can be achieved. On the other hand, due to the rapid growth in textual datasets, for many domains and tasks it is often the case that while a vast amount of textual data is available, only a small portion of the text-instances belong to a specific class of interest (Li et al., 2018; Padurariu and Breaban, 2019; Shaikh et al., 2021).

This sparsity makes annotation challenging because naive data sampling methods do not produce enough examples belonging to a class of interest. It is often possible to use some filtering heuristic before performing manual annotations, so that the percentage of data belonging to the class in question will rise. This approach however is not scalable, since it requires hand-crafting for each specific task or even class. In addition, when filtering is imperfect, the PLM may never observe an important cluster of examples, hurting its generalization capabilities. Therefore, It is essential to minimize the amount of human annotations needed in order to acquire labeled data that will be sufficient for the model to reach a satisfying performance. This calls for the use of Active Learning (AL) techniques. There are many works applying AL techniques for PLM fine-tuning (Gissin and Shalev-Shwartz, 2019; Ash et al., 2020; Dor et al., 2020) but most do not specifically consider the imbalanced dataset case.

Standard AL objectives include obtaining samples which the model is less certain about and increasing the diversity of the chosen sample set. Existing AL methods (C Lin, 2018; Kim and Yoo, 2022) aimed to handle imbalanced datasets suggest doing so by alternating between some approach designed to obtain positive examples[1], and another approach with standard AL objectives. The shortcoming of this alternating approach is that the exact balance of objectives, i.e., how many samples should be chosen aimed to be positive vs diverse, etc., is dependent on the imbalance of the dataset. As a result, existing methods either require a balancing hyperparameter that cannot be tuned in a practical[2] setting, or need to learn the correct balancing ratio over time, making the methods efficient only after a large amount of labelling.

In this work we take a different approach that modifies an AL algorithm to favor positive exam-

[1]Throughout this paper we assume w.l.o.g. that the positive class is the minority class.

[2]AL is most needed when labeled data is missing, therefore assuming a dev set is unpractical.

ples in a way that is self-tuned to obtain an equal amount of positive and negative samples. The core idea of our approach is to debias the inherent skew in the imbalanced dataset by utilizing the inverse-propensity score of the probability of a sample to be positive, according to the model obtained thus far. The simplest form of our algorithm is to sample according to these scores. Intuitively, once the model is reasonable, and this happens quickly in the era of pre-trained models, the amount of positive samples is in the same order of magnitude as the negative samples. We show that these scores can be used not only as a tool to modify the uniform sampling approach, but can be combined with other algorithms providing their own weights to the data points (Yuan et al., 2020), or clustering approaches aimed to maximize diversity (Gissin and Shalev-Shwartz, 2019; Ash et al., 2020).

We demonstrate that for the imbalanced setting, our adaptation of SoTA AL algorithms outperform their original counterparts in multiple datasets. We show that this behavior is consistent across various levels of data imbalance and that in the balanced setting, our adaptation is comparable to the original AL algorithms. The latter property is crucial from a practical perspective as the imbalance ratio of a dataset is rarely known in advance and can be difficult to estimate.

Concluding, we showcase a weakness of recent AL approaches when dealing with imbalanced datasets and devise a novel self-tuned reweighting solution that complements existing (including SoTA) AL algorithms, improving their performance on imbalanced datasets. We experiment on four imbalanced datasets from different tasks and domains. We publish our code and imbalanced datasets for reproducibilty and to encourage future research in this area[3].

2 Related Work

Conventional active learning. There is a myriad of works on active learning in different settings (Ren et al., 2021; Fu et al., 2013). Herein, we consider a pool-based active learning scenario where the algorithm has access to a large unlabeled dataset $\mathcal{U}$ that can be labeled through human annotations. As previously mentioned, our solution modifies existing active learning algorithms in order to better deal with imbalanced datasets. We identify three types of active learning algorithms: random, embedding-based, and score-based algorithms. Random sampling, the simplest approach, generates a distribution over $\mathcal{U}$ from which k distinct samples are randomly selected (e.g., uniform sampling).

Embedding-based algorithms embed unlabeled samples in a high dimensional space. A subset of samples is then selected for annotation using a clustering algorithm based on their embedding, to increase the sample diversity. BADGE (Ash et al., 2020) is an embedding-based algorithm where the sample gradient from the last layer of the model, taken from the log loss of the predicted label, is used as an embedding. ALPS (Gissin and Shalev-Shwartz, 2019) adopts a similar approach, but uses an embedding that captures the language model uncertainty of the different tokens in the sentence.

Score-based algorithms attach a model score to each sample. The most common score-based approach is Least Confidence (Lewis and Gale, 1994) which selects the top-K samples with the highest model uncertainty. A more recent work called DAL (Gissin and Shalev-Shwartz, 2019) trains a classifier to detect new samples dissimilar from the already labeled data.

Active learning in skewed datasets. Compared to active learning, relatively only a few studies have specifically addressed the issue of class imbalance. A common approach in handling class imbalance is to skew the active learning algorithm towards ranking positive samples higher. ODAL (Barata et al., 2021) is an extension of DAL that targets the cold-start setting using an outlier detection algorithm until a single positive sample is found. HAL (Kazerouni et al., 2020) suggests a Hybrid Active Learning algorithm that switches between selecting 'exploitation' that chooses points where the model is uncertain and 'exploration' that either chooses samples uniformly at random or according to how far they are from already chosen samples, yet this approach requires a hyperparameter determining how to balance exploration and exploitation, that is crucial to the performance, and setting it is said to be an open problem.

Recent works try to combine conventional active learning algorithms with positive-skewed sampling. C Lin (2018) explore a method that combines multiple strategies for selecting new examples such as standard active learning, sampling in a skewed way towards positive examples, and generating examples of the minority type (via manual annotators).

[3] https://github.com/balancingAL/ImbAL

They apply a Multi-armed bandit (MAB) algorithm (UCB) to balance between the strategies. Similarly, BMP (Kim and Yoo, 2022) defines two types of policies, one samples from the positive class and the other uses a standard AL algorithm such as random or VE (Beluch et al., 2018), and uses a MAB algorithm to dynamically allocate batches to the different policies. It is folklore knowledge that MAB algorithms require a large number of rounds (at least 10s) before they can learn anything meaningful[4], and indeed the number of rounds in the mentioned papers are 55 or more. Having a large number of rounds can be computationally expensive as SoTA active learning algorithms require applying a large ML model on all examples, as well as time consuming in a human annotation setting. We thus focus on a small number of rounds (say, 5), making these works less applicable.

3 Setting

We consider the pool-based active learning setting, where we are given a pool of unlabeled samples $\mathcal{U}$. At each iteration t, some subset $S_t \subset \mathcal{U}$ of size k is selected for annotation. This subset, sometimes together with $S_1, \ldots, S_{t-1}$, is used to train a model M_t. The subset S_t is selected by a sampling algorithm $\mathcal{A}$, which is usually dependent on the model M_{t-1}. A formal description of this process is given in Algorithm 1. The goal in this setting is to select a subset $\mathcal{D} \subset \mathcal{U}$ to be labeled, such that a model trained on $\mathcal{D}$ optimizes some metric on a separate test set. We focus on the binary classification problem, as it includes multiple common and important imbalanced scenarios such as phishing attempts, fraud, etc. (Kazerouni et al., 2020; Barata et al., 2021; C Lin, 2018).

4 Our approach

Consider some iteration t where we have at our disposal a model M_{t-1} trained on samples labelled in previous iterations.[5] For a sample $x \in \mathcal{U}$, we consider $M_{t-1}(x)$ as an approximation of the probability of x being a positive example. If we utilize the naive random algorithm to select samples $x \in \mathcal{U}$, then $M_{t-1}(x)$ is a random variable in $[0, 1]$. Our goal is to construct a distribution p over $\mathcal{U}$ such that the resulting random variable $M_{t-1}(x)$ is uniformly distributed in $[0, 1]$. To motivate this idea, consider a case where the model is perfectly calibrated, meaning the probability it provides is correct. Here, if sampled according to p, half of the examples, in expectation, will be positive (in Observation 1 we provide an additional bound under a weaker assumption). Another advantage of such a distribution appears even in the case of balanced datasets. Consider a case where many of the examples are easy to classify. It is likely that in an early stage easy samples will be concentrated near 0 and 1 in the distribution of $M_{t-1}(x)$. This means that our sample strategy will upweight exactly the problematic points where the model is uncertain, such that we obtain the property of biasing towards informative examples.

Algorithm 1 Pool-based Active Learning

Input: Unlabeled data pool $\mathcal{U}$, number of samples per iteration k, number of iterations T, sampling algorithm $\mathcal{A}$, classifier model training algorithm $\mathcal{M}$ and seed samples $\mathcal{S}$.

$\mathcal{D} \leftarrow \mathcal{S}$
$M_0 = \mathcal{M}(\mathcal{D})$ ▷ Initiate classifier model
for $t \leftarrow 1 : T$ **do**
  $S_t = \mathcal{A}(M_{t-1}, \mathcal{U}, k)$ ▷ Sampling step
  Annotate S_t
  $\mathcal{U} \leftarrow \mathcal{U} \setminus S_t$
  $\mathcal{D} \leftarrow \mathcal{D} \cup S_t$
  $M_t = \mathcal{M}(\mathcal{D})$ ▷ Learning step
Return M_T

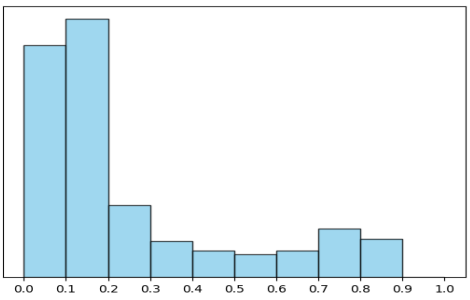
The distribution p described above can be achieved by weighting each sample inversely to its propensity results in a uniform sample. Thus, we call this method **InvProp**. In terms of implementation, we discretize the interval $[0, 1]$ into m bins uniformly, meaning $[0, \frac{1}{m}), [\frac{1}{m}, \frac{2}{m}), \ldots, [\frac{m-1}{m}, 1]$, as can be seen in Figure 1(a). For bin i that contains b_i samples we give each sample a probability proportional to $\frac{1}{b_i}$ (see Figure 1(b)).

The following lemma demonstrates the robustness of InvProp-weights to the exact, and unknown, degree of imbalance in the data.

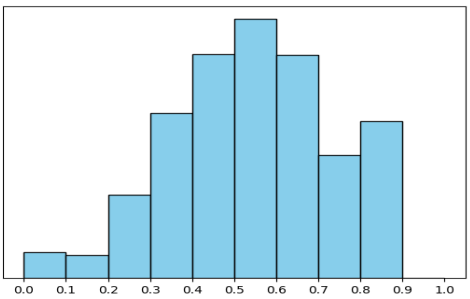
Lemma 1. *Let $\mathcal{A}$ be the random algorithm with InvProp weights defined over 2 bins, and let M be a model with precision α. Also, let $S = \mathcal{A}(M, \mathcal{U}, k)$*

[4] This follows from a simple statistics exercise showing that with a handful of rounds, the posterior distributions of the reward of each option is very close to the prior, or that the confidence intervals remain large.

[5] We assume that the process begins with a model trained on a handful of examples. This is a realistic setting, as it is rarely the case that a practitioner will aim to solve a classification problem without having a single labelled example. Alternatively, we can begin with uniform weights. We discuss this issue in Section 5.9.



(a) Histogram of confidence scores $M_t(x)$.



(b) Inverse propensity distribution p.

Figure 1: Confidence scores histogram and the probability fitting the InvProp-weights.

be the subset of size k sampled by $\mathcal{A}$. If the number of samples in bin $[1/2, 1]$ is at least $\beta \cdot k$ for $\beta > 0$, then the expected number of positive samples in S is $\mathbb{E}\left[\left|\{s \in S \mid y_s = 1\}\right|\right] \geq \frac{\alpha\cdot\gamma\cdot k}{2}$, where $\gamma = \beta/2$ if $\beta < 1$ and $\gamma = 1 - \frac{1}{2\beta}$ if $\beta \geq 1$.

The proof for the lemma can be found in Appendix A. The lemma does not directly translate to the setting with $m > 2$ bins, but by altering the precision assumption with that of the model being approximately calibrated, we keep the guarantee of obtaining a constant fraction of positive examples[6]. In our ablation studies (see section 5.8) we see that the precise number of bins has a small effect on the result, resulting in our decision to use an arbitrary default value (of $m = 10$) in our experiments.

4.1 Extending to SoTA algorithms

The random algorithm combined with the InvProp distribution, as described above, enjoys being skewed towards the minority class and informative samples. Nevertheless, recent AL algorithms have been able to provide additional benefit over the naive random approach. This leads us to incorporate our weighting scheme in SoTA AL algorithms. To this end, we consider the underlying distribution p of the InvProp method as a weighting function, assigning each point x a weight denoted by $w(x)$ (in the section above this is the probability of choosing x). For score based algorithms that choose samples based on a score $s(x)$, we modify the score to become $s_w(x) = w(x) \cdot s(x)$ (see Algorithm 2). For embedding-based algorithms that choose samples based on clustering, we modify the clustering procedure to be weighted according to $w(x)$ (see Algorithm 3). With this strategy, we are able to modify SoTA algorithms such as BADGE (Ash et al., 2020), ALPS (Yuan et al., 2020) and DAL (Gissin and Shalev-Shwartz, 2019) to be better suited to imbalanced datasets.

Algorithm 2 Weighted Score-based AL

Input: Unlabeled data pool $\mathcal{U}$, number of samples to select k, scoring function s, weight function w.

$s_w(x) = \sum_{x\in S} w(x) \cdot s(x)$

Return $\text{argmax}_{S\subseteq\mathcal{U}} \left\{ s_w(x) \;\middle|\; |S| = k \right\}$

Algorithm 3 Weighted Embedding-based AL

Input: Unlabeled data pool $\mathcal{U}$, number of samples to select k, embeddings e_x, weight function w.

Cluster $\{e_x\}_{x\in\mathcal{U}}$ with weights $w(x)$ into k clusters, pick S to be the samples nearest to the cluster centers.

Return S

5 Experiments

5.1 Datasets & Measure

We follow recent AL work (Ein-Dor et al., 2020; Yuan et al., 2020; Wang et al., 2021) and use 4 popular text classification datasets. **DBpedia-14** (Zhang et al., 2015) contains text snippets from 14 different DBpedia (Lehmann et al., 2015) classes. Each snippet's label is the ontology class it was taken from. **SST-2** (Socher et al., 2013) is a binary classification dataset containing sentences from movie reviews labeled by their sentiment. **PubMed-20K-RCT** (Dernoncourt and Lee, 2017) contains sentences from PubMed abstracts labeled by their role in the abstract. **AG_news** (Zhang et al., 2015) contains news snippets classified into 4 categories.

Binary-Imbalanced data. We generate several Binary-Imbalanced versions from each of the above

[6]The precise assumption is that for a bin $[\rho, \rho + 1/m]$, at least $\alpha\rho$ of the samples have a positive label. This property measures how calibrated the model is; indeed a perfectly calibrated model guarantee this with $\alpha = 1$. In this case we are guaranteed that in expectation, $\sum_{i=0}^{m-1} \frac{i\alpha\gamma k}{2m^2} = \frac{(m-1)}{4m}\alpha\gamma k \geq \alpha\gamma k/6$ samples are positive.

mentioned datasets. This is done by first selecting a class from the original dataset to serve as the positive (minority) class in the generated dataset. Samples from all other classes are regarded as negative. Both train and test sets are updated this way. For DBpedia-14, due to its larger number of classes and small variance we observe on this dataset, we randomly select 3 of the classes to serve as the minority class. For the rest of the datasets we use each of the original classes as a minority class. After selecting a positive class and updating the train and test sets, we sample positive examples from the training set so that the fraction of positive samples in the resulting dataset will be equal to the desired value. We construct datasets where the fraction of positives is $1/x$ for $x \in \{2, 10, 20, 50, 100\}$, i.e., 4 different imbalance ratios and a balanced dataset. See Table 1 for further details about the datasets.

Measure. As customary in studies related to imbalanced classes, we measure Balanced Accuracy and ROC-AUC, metrics that are insensitive to change in class distribution. Thus, the test set is not down-sampled, as this would only add noise to the evaluation. In addition, we compare the Positives Ratio, the fraction of minority samples found by each algorithm. This will allow us to evaluate the skew produced by our technique. Due to aggregations over different datasets, we found the standard deviations tend to be large, making confidence intervals uninformative. We thus measure statistical significance via p-values[7] and mark in **bold** results whose p-value is smaller than 5% for all compared results.

5.2 Active Learning Algorithms

We test our proposed weighting method by incorporating it into well established AL algorithms.

- **BADGE** (Ash et al., 2020) is an embedding-based algorithm. It generates a weak label according to the prediction of the current model. The weak label is used to calculate the gradient of the last layer of the model. The gradients are used as the embeddings.
- **ALPS.** (Yuan et al., 2020) Is an embedding-based algorithm. It generates an embedding according to the MLM objective. Some tokens are masked and predicted by the model. The embeddings are defined as the cross entropy distance between the prediction of the model and the actual masked tokens.
- **DAL.** (Gissin and Shalev-Shwartz, 2019) Is a score-based algorithm. It trains a classification model to predict whether a sample is labelled or unlabelled. The "top" unlabelled samples (the samples with the highest confidence score) are selected.
- **Random.** The naive random baseline which selects samples uniformly at random.

5.3 Weighting Methods

Other than our main approach of **InvProp**, we evaluate two additional weighting methods.

- **Uniform** ignores the imbalance issue and assigns the same weight to all samples, effectively running the original AL algorithm.
- **PosProb** assigns a point x the weight $M_{t-1}(x)$, meaning the probability according to the available model that the sample is positive. This option is explored as a strawman that promotes positive samples, but is not adaptive to the imbalance ratio.

We test 3 weighting methods (Uniform, PosProb, InvProp), incorporated into the 4 mentioned AL algorithms. Overall, 12 procedures are tested.

Previous studies came to the conclusion that even though AL algorithms improve over naive random sampling, there is no single algorithm which performs best for all models and tasks (Lowell et al., 2019; Dor et al., 2020). Hence, we do not aim to show that one algorithm outperforms another, we aim to test whether utilizing our weighting method improves the performance of each of the base algorithms when applied on imbalanced datasets.

5.4 Experimental Setup

We evaluate each algorithm on all Binary-Imbalanced datasets through 10 runs, each with a different random seed that determines the shuffling and random initialization. The results of these runs were averaged to produce the reported results. We ran the algorithms for 5 AL iterations, selecting 50 samples at each iteration.

We use the uncased bert-base model with 110M parameters[8] (Devlin et al., 2019) as the PLM, and set the maximum sequence length to 128 and the number of train epochs to 2. To cope with the imbalance of the data that was already labeled, we use oversampling to ensure that the positive samples

[7] The p-value is calculated using a relative t-test of the elements whose mean is calculated.

[8] https://huggingface.co/transformers/

Dataset	Classes	Original train/test	Selected minority classes
DBpedia-14	14	$560K/70K$	Company, Animal, Album
SST-2	2	$\sim 67K/872$	all classes
Pubmed-20K-RCT	5	$\sim 180K/30K$	all classes
AG_news	4	$120K/7.6K$	all classes

Table 1: **Dataset details.** Number of classes and train/test sizes in the original dataset, and the set of classes from the original dataset used as the minority class in the Binary-Imbalanced datasets we created (see section 5.1).

are at least a quarter of the training samples. This heuristic is known to vastly reduce the negative effects of the imbalance (Estabrooks and Japkowicz, 2001). At each iteration, we fine-tune the original pre-trained model using all examples, rather than fine-tuning the resulting model of a previous iteration, as this is known to provide better performance (Ash and Adams, 2020).

At the beginning of each run, we generate a random labeled subset of 5 negative and 5 positive samples that is given to the algorithm. An intuitive design would be that of a 'cold start', beginning with no labeled data. However, we found this setup to have a very high variance in terms of the final performance. In Section 5.9 we elaborate on the conjectured reason for this, provide experimental results to support our claim, and motivate our choice of starting with 5 examples of each class.

5.5 Main Results

In order to concisely compare between the weighting methods, we aggregate the results over either algorithms or datasets. Table 2 compares the performance of the InvProp and Uniform weighting method on the 4 datasets. A detailed comparison to PosProb can be found in Section 5.6. For each Binary-Imbalanced dataset and weighting method, we select the algorithm that performed best for the combination, based on its average score over 10 random seeds. We then take the average metric over Binary-Imbalanced variants of a dataset to obtain a single score for each (metric, dataset, weighting procedure) triplet. Here, we consider only Binary-Imbalanced datasets with an imbalance ratio of $\frac{1}{50}$. In both AUC and Balanced Accuracy, InvProp provides either superior or comparable results. As a justification, there is an overwhelming advantage to our weighting method in terms of positive ratio, nearly doubling the number of positive examples.

In Table 3 we compare the performance of the weighting methods between different AL algorithms. For each algorithm we average the score obtained by a specific weighting method over all

	Balanced Accuracy		AUC		Positives Ratio	
	InvProp	Uniform	InvProp	Uniform	InvProp	Uniform
AG_news	**0.7889**	0.7674	**0.9475**	0.9414	**0.3601**	0.2030
DBpedia	**0.9876**	0.9826	0.9983	0.9983	**0.4377**	0.2287
Pubmed	0.6338	0.6387	**0.8813**	0.8606	**0.2516**	0.1217
SST-2	0.6523	0.6299	**0.8709**	0.8538	**0.2045**	0.0989

Table 2: Comparison of InvProp with Uniform across multiple datasets (for an imbalance ratio of $\frac{1}{50}$).

	Balanced Accuracy		AUC		Positives Ratio	
	InvProp	Uniform	InvProp	Uniform	InvProp	Uniform
ALPS	**0.7002**	0.6887	0.8712	0.8625	**0.0368**	0.0271
BADGE	0.7203	0.7181	0.8914	0.8901	**0.2730**	0.1646
DAL	**0.7036**	0.6692	**0.8948**	0.8560	**0.2823**	0.0142
Random	**0.7248**	0.6763	**0.8986**	0.8681	**0.2442**	0.0196

Table 3: Comparison of InvProp and Uniform across multiple base algorithms (for an imbalance ratio of $\frac{1}{50}$).

Binary-Imbalanced datasets, averaged on 10 random seeds. Here we also consider an imbalance ratio of $\frac{1}{50}$. For both AUC and Balanced Accuracy we see a clear advantage to InvProp-weighting compared to Uniform weighting, in that it is either better or comparable for all algorithms. In terms of the Positives Ratio, the effect on the different algorithms varies, but we see a clear increase when using the InvProp weighting method. The method is able to produce more positives, and as shown in Appendix B is able to do this throughout the different iterations.

Two interesting insights that are not directly related to our study are (1) the algorithms differ in their robustness to imbalance, with BADGE proving to be quite robust, especially when Uniform weighting is applied; (2) the Random algorithm combined with our weighting method becomes a strong baseline and is in fact the leader w.r.t both Balanced Accuracy and AUC.

We ran a similar comparison against the PosProb baseline and concluded that for an imbalance ratio of $\frac{1}{50}$, the results are comparable. We provide the full experiments comparing all 3 weighting methods on multiple imbalance ratio combinations, on all datasets and base algorithms in Section 5.6. In

	Balanced Accuracy		AUC		Positives Ratio	
	InvProp	Uniform	InvProp	Uniform	InvProp	Uniform
2.0	0.9009	0.9029	0.9518	**0.9534**	0.4816	0.4855
10.0	**0.8253**	0.8149	**0.9348**	0.9306	**0.3259**	0.1436
20.0	**0.7709**	0.7519	**0.9175**	0.9061	**0.2779**	0.0946
50.0	**0.7122**	0.6881	**0.8890**	0.8692	**0.2091**	0.0564
100.0	**0.6839**	0.6612	**0.8552**	0.8454	**0.1692**	0.0375

Table 4: Comparison of InvProp with Uniform across multiple degrees of imbalance.

	Balanced Accuracy		AUC		Positives Ratio	
	InvProp	PosProb	InvProp	PosProb	InvProp	PosProb
2.0	**0.9009**	0.8313	**0.9518**	0.9442	0.4816	**0.6425**
10.0	**0.8253**	0.8155	**0.9348**	0.9310	0.3259	**0.3702**
20.0	0.7709	0.7744	**0.9175**	0.9140	0.2779	0.2821
50.0	0.7122	0.7128	**0.8890**	0.8833	**0.2091**	0.1831
100.0	0.6839	0.6800	0.8552	0.8513	**0.1692**	0.1273

Table 5: Comparison of InvProp with PosProb across multiple degrees of imbalance.

addition, a qualitative analysis of the informativeness of selected positive samples can be found in Appendix C.

5.6 Sensitivity to Imbalance Ratio

The main approach of techniques addressing dataset imbalance is to skew the sampling process towards positive samples. This leads to the question of when each technique should be applied; optimizing towards positive samples may have a negative effect on the results when the real-world distribution is almost balanced. Therefore, we evaluate the performance of InvProp-weighting on datasets with different imbalance ratios. As described in Section 5.1, we generated datasets with different imbalance ratios. Specifically, we use datasets with an imbalance ratio of $100, 50, 20, 10$ and 2 (i.e. balanced). As in earlier experiments, we repeat the experiment on each Binary-Imbalanced dataset with 10 different random seeds. For each imbalance ratio we compare the average performance of each weighting method over all AL algorithms and datasets (the results per dataset and algorithm appear in Appendix D). Results of this experiment are reported in Table 4; it is easy to see that the InvProp-weighting method outperforms the Uniform baseline across all imbalance ratios which represent an imbalanced dataset. In addition, even for the balanced scenario, the baseline slightly outperforms our method in only one metric, AUC, while being comparable in the Balanced Accuracy metric and in Positives Ratio. This allows us to recommend the usage of InvProp-weighting even if the imbalance ratio is unknown.

As mentioned in Section 5.5, the PosProb weighting method performance is comparable to InvProp when considering a dataset with an imbalance ratio of $\frac{1}{50}$. In Table 5 we extend this comparison and compare the effect of the imbalance ratio on these two weighting methods. This shows that InvProp either outperforms or is comparable to PosProb in both Balanced Accuracy and AUC. A deeper dive into the Balanced Accuracy scores shows that PosProb performs best (compared to InvProp) on datasets with an imbalance ratio of 20. Since the weights of PosProb are independent of the imbalance of the dataset, i.e., down-sampling the minority class does not affect the proportion between weights of two of the remaining samples, it is expected that the performance of PosProb will peak at some specific imbalance ratio and decline as the imbalance ratio changes further away from this peak.

Another important aspect of the results is the number of positive samples accumulated by each solution. For the balanced dataset, PosProb overskews the selection toward the positive class, leading to more than 60% of positive selected samples. As the imbalance grows, this stabilizes, and for imbalance ratio of 50 and 100 InvProp is able to identify a larger number of positive samples. The higher AUC score of InvProp points toward PosProb finding less diverse positive samples due to the large weight it assigns to positive samples which the model is certain about.

5.7 Extreme imbalance

To further test the limits of our methods, we extend the imbalance ratios to 200 and 1000. Results are presented in Table 6. Interestingly, even though the InvProp and PosProb weighting schemes manage to produce more positive examples than the uniform scheme, the Balanced Accuracy and the AUC metrics are better for the uniform as the imbalance ratio reaches extremes such as 1000. This could be explained by the weighting methods choosing positive examples that are too similar which do not contribute to the efficacy of the model. In general, we believe that in such extreme cases, where positive examples are so scarce, the trained model is unable to learn and is under-performing, hindering the efficacy of any AL approach. The performance of all AL algorithms with the Uniform weighting scheme exemplifies this, where we see that random

	Balanced Accuracy			AUC			Positives Ratio		
	InvProp	PosProb	Uniform	InvProp	PosProb	Uniform	InvProp	PosProb	Uniform
200.0	**0.6619**	0.6543	0.6377	0.8244	0.8251	**0.8302**	**0.1263**	0.0886	0.0267
1000.0	0.6181	0.6154	0.6160	0.7821	0.7831	**0.8081**	**0.0535**	0.0304	0.0103

Table 6: Comparing the weighting schemes against extreme imbalance ratios.

sampling produces the highest AUC score all of the AL algorithms, with statistical significance (see Table 9, Appendix D).

5.8 Sensitivity to number of bins

One drawback of InvProp-weighting compared to Uniform and PosProb-weighting is the addition of a new parameter, the number of bins. We evaluate the effect of the number of bins on the performance of different AL algorithms (with InvProp-weights) on the Binary-Imbalanced variants of the AG_news dataset. We consider InvProp with $2, 5, 10, 15$ and 20 bins. Results appear in table 11 in Appendix D. In table 12 (Appendix D), we evaluate the sensitivity to the number of bins by measuring significance with a relative t-test. This table lists the p-value $p(m)$ of a test comparing AUC and Balanced Accuracy for the selection of m bins (for $m \in \{2, 5, 15, 20\}$) to that of 10, as 10 was our choice throughout the paper. The results demonstrate that our method is insensitive to the bin count hyper-parameter, as except for one outlier, all p-values fall in $[0.05, 0.95]$. This motivates our choice to avoid optimizing this parameter, thus fixing it to a single value throughout the paper. This experiment was performed for all imbalance ratios discussed in this paper $(2, 10, 20, 50, 100)$. For brevity, we report only on an imbalance ratio of 100, since for this value the affect of the number of bins parameter was the largest. Since we conclude that this parameter is not significant even for this imbalance ratio, this conclusion is relevant across all imbalance ratios.

5.9 Warm start

Recall that in lieu of starting the learning process with no examples (cold-start), we start the process with 5 examples from each class. In Table 10 (Appendix) we show the size of the confidence intervals for the BADGE algorithm over the different datasets, showing an increase of at least $1.5X$, sometimes over $10X$ when comparing the cold-start to the chosen warm-start scenario. A major cause of the cold-start variance is the time it takes the learner to achieve a handful of positive examples. Indeed, when the imbalance ratio is 50, a random selection is expected to obtain only 1 positive sample in each iteration, and in many runs the learner fails simply as it does not find a sufficient initial amount of positive examples. A full solution should discuss realistic methods for obtaining an initial seed of examples, but this challenge is outside the scope of this paper. In order to remove the noise in the evaluation process originating from the time it takes to obtain a handful of examples, we have all methods initialize with the mentioned warm-start. We chose to work with particularly 5 examples as this is a small enough number of samples for it to be easy to obtain even in the imbalance scenario, and on the other hand, the variance is significantly reduced compared to the 'cold-start' setting.

6 Conclusions

In this paper, we tackled the problem of AL in imbalanced datasets. We propose a novel weighting-technique, InvProp , and apply it to three recent AL algorithms (BADGE, APLS and DAL) as well as to naive random sampling. We show results for the PosProb weighting scheme in addition to InvProp and compare to a uniform weighting baseline on four datasets. We show that InvProp-weighting consistently finds more positive examples, and leads to better or comparable performance compared to other weighting schemes on all tested datasets and AL algorithms. We also test various imbalance settings showing that InvProp outperforms the Uniform baseline across all imbalance ratios which represent some imbalance. In addition, on a balanced dataset the Uniform baseline outperforms our method in only one metric, while being comparable in the rest. Compared to PosProb, our solution proved to be more robust to changes in the imbalance ratio. When testing extreme scenarios where the imbalance is less than 0.5%, all tested algorithms perform worse than the simple random sampling baseline, suggesting that AL algorithms are not advantageous given an uninformative model. Concluding, we show our novel weighting scheme improves several SoTA AL algorithms on various

datasets and imbalance ratios.

References

Jordan Ash and Ryan P Adams. 2020. On warm-starting neural network training. *Advances in Neural Information Processing Systems*, 33:3884–3894.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ricardo Barata, Miguel Leite, Ricardo Pacheco, Marco OP Sampaio, João Tiago Ascensão, and Pedro Bizarro. 2021. Active learning for imbalanced data under cold start. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9.

William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. 2018. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9368–9377.

Mausam C Lin. 2018. Active learning with unbalanced classes & example-generated queries. In *AAAI Conference on Human Computation*.

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k RCT: a dataset for sequential sentence classification in medical abstracts. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pages 308–313. Asian Federation of Natural Language Processing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Liat Ein Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active learning for bert: an empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active learning for BERT: an empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7949–7962. Association for Computational Linguistics.

Andrew Estabrooks and Nathalie Japkowicz. 2001. A mixture-of-experts framework for learning from imbalanced data sets. In *International Symposium on Intelligent Data Analysis*, pages 34–43. Springer.

Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. *Knowledge and information systems*, 35(2):249–283.

Daniel Gissin and Shai Shalev-Shwartz. 2019. Discriminative active learning. *CoRR*, abs/1907.06347.

Abbas Kazerouni, Qi Zhao, Jing Xie, Sandeep Tata, and Marc Najork. 2020. Active learning for skewed data sets. *arXiv preprint arXiv:2005.11442*.

Gwangsu Kim and Chang D Yoo. 2022. Blending query strategy of active learning for imbalanced data. *IEEE Access*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Yijing Li, Haixiang Guo, Qingpeng Zhang, Mingyun Gu, and Jianying Yang. 2018. Imbalanced text sentiment classification using universal and domain-specific knowledge. *Knowl. Based Syst.*, 160:1–15.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

David Lowell, Zachary C Lipton, and Byron C Wallace. 2019. Practical obstacles to deploying active learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 21–30.

Cristian Padurariu and Mihaela Elena Breaban. 2019. Dealing with data imbalance in text classification. In *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES-2019, Budapest, Hungary, 4-6 September 2019*, volume 159 of *Procedia Computer Science*, pages 736–745. Elsevier.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke

Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. 2021. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40.

Sarang Shaikh, Sher Muhammad Daudpota, Ali Shariq Imran, and Zenun Kastrati. 2021. Towards improved classification accuracy on highly imbalanced text dataset using deep neural language models. *Applied Sciences*, 11(2).

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 4195–4205. Association for Computational Linguistics.

Michelle Yuan, Hsuan-Tien Lin, and Jordan L. Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7935–7948. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

A Proof of bound on expected number of positives samples

Proof of Lemma 1. Let M be a model with precision α and let $\mathcal{A}$ be the random algorithm with InvProp weights defined over 2 bins. Consider a split of all unlabeled samples into two bins according to the score returned by M. In order to bound the number of positive samples selected by $\mathcal{A}$ we first bound the number of samples selected from bin b_2, that is, the number of samples with a score in the range $[1/2, 1]$.

In the first iteration of the random selection process the probability of selecting a sample from b_2 is exacty $\frac{1}{2}$. After each iteration where a sample is selected from b_2 the probability decreases as the total weight of samples in b_2 becomes smaller. For example, if at the first iteration a sample is selected from b_2, the probability of selecting another sample from b_2 is at least $\frac{k-\frac{1}{\beta}}{2k-\frac{1}{\beta}}$, as initially there are at least k samples in b_2. Thus, we will bound the expected number of samples select from b_2 at iteration i by calculating the probability of selecting such a sample assuming all previous samples were selected from b_2.

We denote the sample selected at iteration i by s_i and the set of samples selected until iteration i by S_i. Thus, we get that

$$\begin{aligned}\sum_{i=1}^{\min\{\beta\cdot k,k\}} \Pr[s_i \in b_2 \mid S_i] = \\ = \sum_{i=1}^{\min\{\beta\cdot k,k\}} \frac{\sum_{s\in b_2\setminus S_i} w_s}{\sum_{s\in\mathcal{U}\setminus S_i} w_s} \geq \\ \geq \sum_{i=1}^{\min\{\beta\cdot k,k\}} \frac{\sum_{s\in b_2} w_s - \frac{|S_i|}{\beta k}}{\sum_{s\in\mathcal{U}} w_s - \frac{|S_i|}{\beta k}} = \\ = \sum_{i=1}^{\min\{\beta\cdot k,k\}} \frac{k-\frac{i-1}{\beta}}{2k-\frac{i-1}{\beta}} \qquad (1)\end{aligned}$$

If $\beta < 1$ then we get that Equation 1 is equal to

$$\begin{aligned}\sum_{i=1}^{\beta\cdot k} \frac{k-\frac{i-1}{\beta}}{2k-\frac{i-1}{\beta}} \geq \beta\cdot\sum_{i=1}^{k}\frac{k-i+1}{2k-i+1} \geq \\ \geq \frac{\beta}{2k}\sum_{i=1}^{k}(k-i+1) > \\ > \frac{\beta}{2k}\cdot\frac{k^2}{2} = \frac{\beta k}{4}.\end{aligned}$$

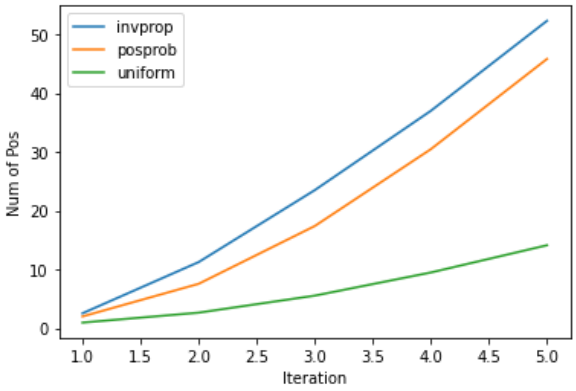


Figure 2: Number of positive results found by each weighting scheme over time for an imbalance ratio of 50, averaged over all datasets and all base algorithms.

Otherwise, if $\beta \geq 1$ then Equation 1 is equal to

$$\begin{aligned}\sum_{i=1}^{\min\{\beta\cdot k,k\}} \frac{k-\frac{i-1}{\beta}}{2k-\frac{i-1}{\beta}} \geq \sum_{i=1}^{k}\frac{k-\frac{i-1}{\beta}}{2k} \geq \\ \geq \frac{k}{2} - \frac{\frac{k^2}{2\beta}}{2k} = \\ = \frac{k}{2}(1-\frac{1}{2\beta}).\end{aligned}$$

We can conclude that the expected number of samples selected from bin b_2 is $\frac{\gamma\cdot k}{2}$, where $\gamma = \beta/2$ if $\beta < 1$ and $\gamma = 1 - \frac{1}{2\beta}$ if $\beta \geq 1$.

It is important to note that the sampling process does not differentiate between samples in the same bin as they have the same weight. Thus, we can assume that the samples selected from b_2 are selected uniformly at random. Since the precision of the model is α, the expected probability of a selected sample to be positive is α as well. Thus, the expected number of positive samples selected from b_2 must be at least $\frac{\alpha\cdot\gamma\cdot k}{2}$. □

B Balancing the minority class over time

In Section 4 we presented InvProp-weights as a technique for balancing the confidence scores sampled by an AL algorithm. As presented in sections 5.5 and 5.6, utilizing InvProp-weights leads to the selection of more or comparable number of positive samples. In this section we extend this analysis and compare the ability of the different weighting schemes to balance the two classes over time, i.e., over the five iterations.

Figure 2 presents the number of positive samples found by each weighting scheme. We see that at the first iteration, all algorithms struggle with

generalizing over the seed of positive samples and find only a handful of positives. From the second iteration, InvProp dominates all algorithms in the number of positives it finds which we attribute to the inherent exploration of the algorithm. PosProb follows up with a lower number of positive samples while uniform struggles to find positive samples without any skew toward them, as expected from the large imbalance in the data.

C Qualitative analysis of sample informativeness

In Section B it was established that utlizing InvProp better balances the sampled dataset over time. In this section, we examine the informativeness of the selected positive samples by considering the confidence scores of selected samples to see which weighting scheme produces positive samples that the model is uncertain about. Note, more informative samples tend to improve the quality of future models and prevent overfitting. In order to do so we turn to a qualitative analysis. The evaluation is presented for an experiment on the AG_news dataset with an imbalance ratio of 20 using the Random algorithm with both InvProp and PosProb-weights (complementary parameters are as defined by the setting in Section 5.4).

In Table 7 we compare the expected confidence score of a sample selected according to the fitting distribution (i.e., the distribution generated according to InvProp and PosProb respectively). Note, the goal is to select positive samples with low confidence score. Even though the expected confidence score of PosProb is slightly lower at the first iteration, as early as the second iteration the opposite is true. While the expected confidence score, as sampled by InvProp , grows between iterations, this is at a slow pace, showing it is still able to find positive samples that are hard and contribute new knowledge to model at later iterations. On the other hand, the skew of PosProb toward positive samples becomes overwhelming very fast, leading to the vast selection of positive samples the model is practically certain of their label at the last iteration.

Finally, we dive deeper, examining the actual confidence scores, distribution and expected sample generated by the two weighting schemes. For each weighting scheme and each iteration we present three plots that are discretized to simplify the presentation (the confidence scores are partitioned into 10 bins of width 0.1):

Iteration	InvProp	PosProb
1	0.695	0.649
2	0.576	0.586
3	0.647	0.836
4	0.636	0.868
5	0.688	0.904

Table 7: Comparing the expected confidence score of a sample that is sampled according to the distribution generated using InvProp and PosProb-weights at each iteration.

- The confidence scores histogram of the unlabeled samples, describing the portion of samples that fell into each bin.
- A probability function describing the sampling probability of samples as a function of their confidence score.
- Given the probability function used for sampling, the distribution of a sample over the confidence bins, that is, for each bin it describes the probability that a sample belonging to it is selected. In addition, each bin is partitioned into two, showing the portion of positive and negative samples in the bin.

Figure 3 presents the plots when utilizing InvProp-weights to generate the sampling probability, while the plots for PosProb-weights appear in Figure 4.

The plots in Figure 3 suggest that InvProp has two important properties. First, as seen before, it is able to skew the random selection towards selecting a more balanced set of samples, containing nearly equal number of positive and negative samples. This is achieved by attaching higher weights to bins corresponding to higher confidence scores. These, as appear in the plots, tend to contain a higher portion of positive samples, as can be seen through all five iterations. Second, as the set of labeled samples grows and the model provide better performance and produces more accurate results, there are less samples the model is uncertain of (i.e., scores closer to $\frac{1}{2}$). In turn, the InvProp-weightattached to these sample is larger, increasing their sampling probability, as is evident in the plots of the last three iterations. This automatic transition from skewing the selection towards positive samples to focusing on uncertain samples is a desired property in any AL algorithm when dealing with imbalanced datasets.

In contrast, utilizing PosProb-weightleads to a

different behaviour, as can be seen in Figure 4. On the contrary, instead of transitioning to focus on the uncertainty of selected samples, the PosProb-weights skews even more the selection towards positive samples. Looking at the plots for the last three iterations shows that the probability generated when utilizing PosProb almost ignores samples with high uncertainty. This is expected due to the high weight of positive samples and the structure of the confidence scores histograms of mature models. In addition, the large imbalance of the datasets results in many samples being placed in the left most bin (that is, $[0.0, 0.1)$). Thus, the inability of PosProb to dynamically fit its weights to the imbalance leads to the selection of a large portion of negative samples on which the model is already certain about. This is, of course, a waste of resources.

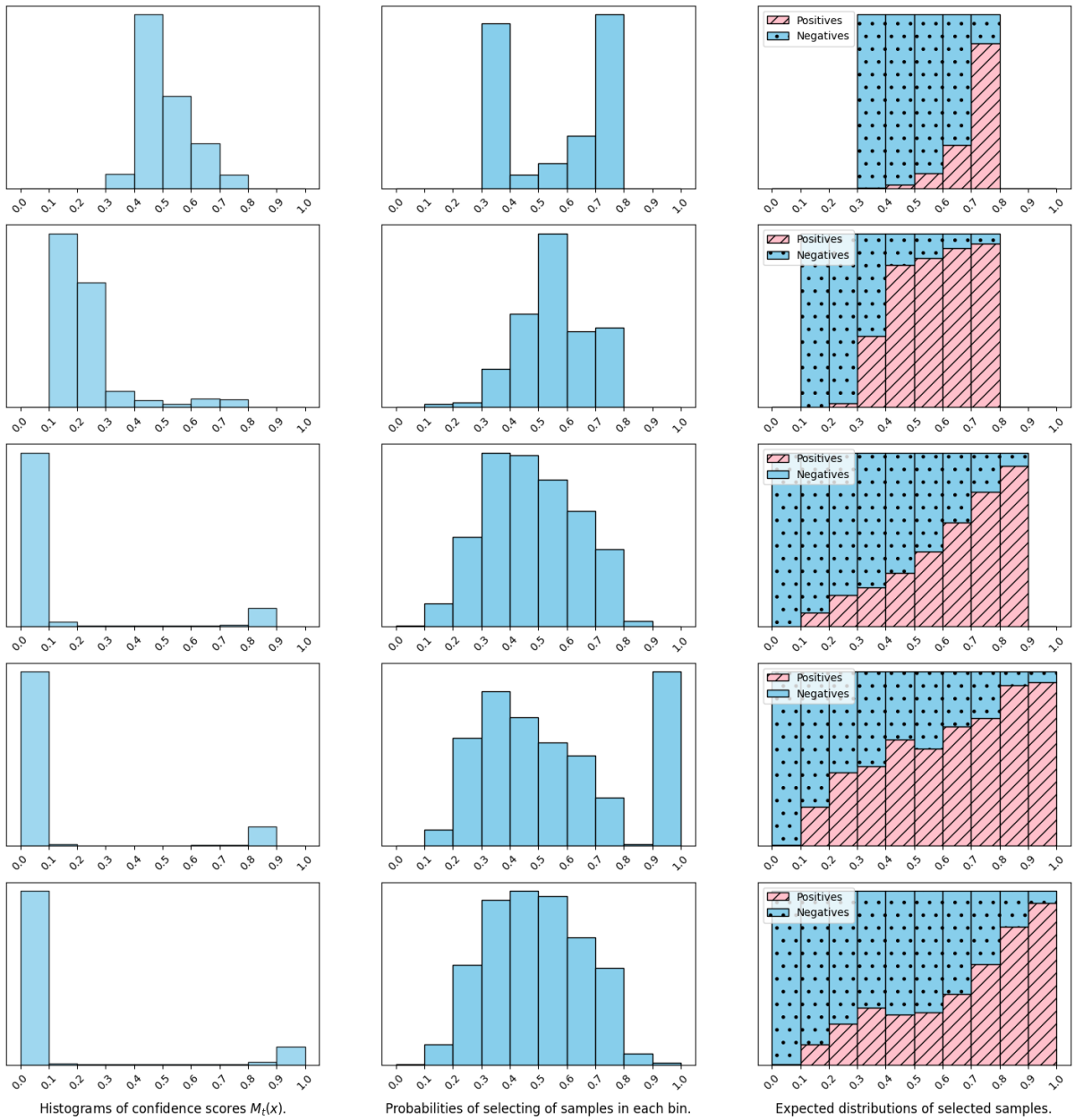


Figure 3: Sampling according to InvProp-weights. The rows represent the five sampling iterations (top row is the first iteration and the bottom row is the last iteration. The left column presents the confidence scores histogram, that is, each bar presents the portion of samples for which the model M_t (where t is the iteration number) gives a score that falls in the confidence interval of the bar. Next, the middle column presents the sampling probability of samples in each bin. Using InvProp the sampling probability of a sample in bin b_i is proportional to $\frac{1}{|b_i|}$. Lastly, the right column presents the probability that a sample sampled according to the fitting distribution (appearing in the middle column) would belong to each confidence score interval. Here the probability over of all intervals is uniform due to the design of InvProp . In addition, each bar in the left column presents the portion of samples that are negative and positive in each confidence interval.

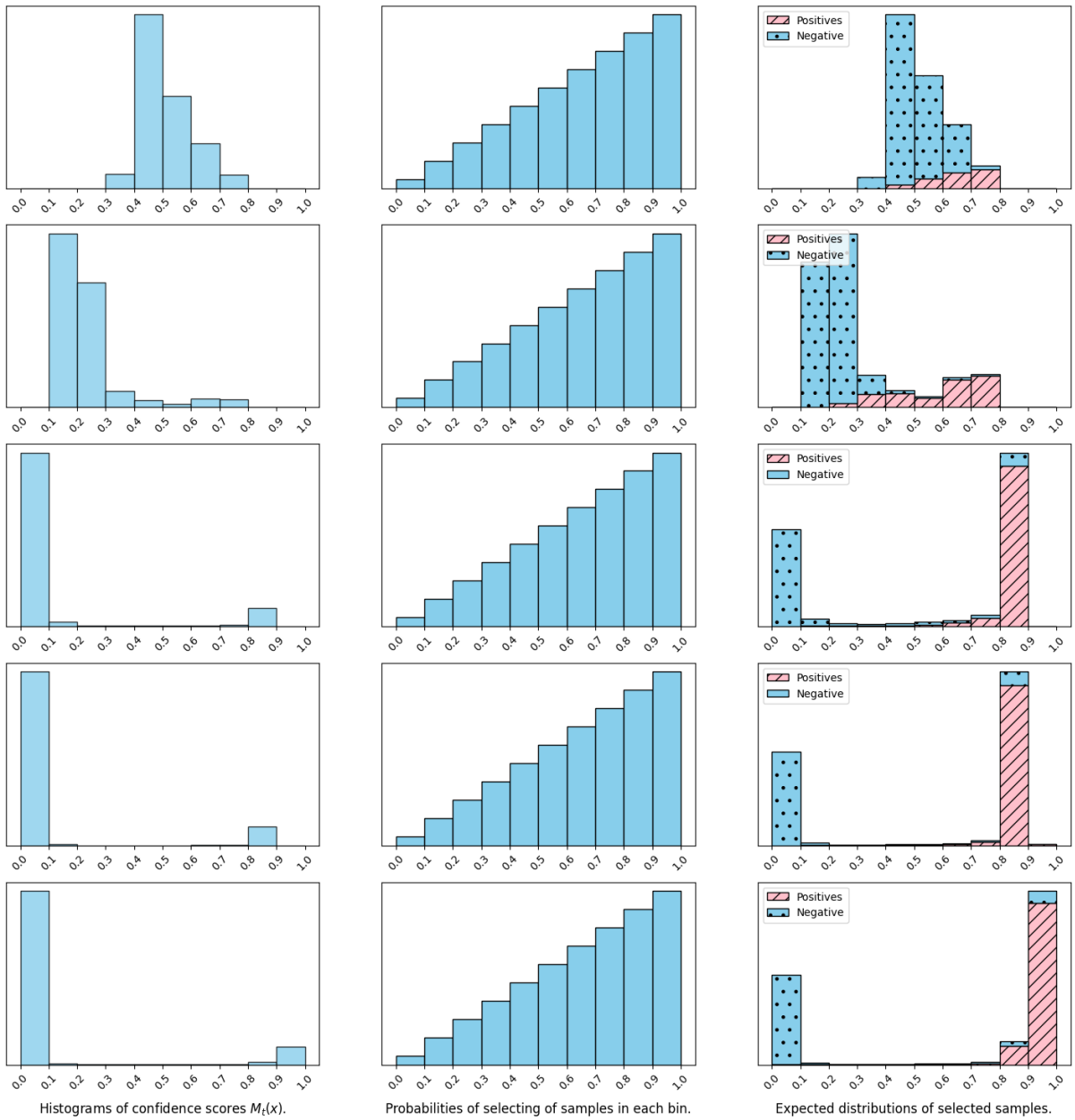


Figure 4: Sampling according to PosProb-weights. The rows represent the five sampling iterations (top row is the first iteration and the bottom row is the last iteration. The left column presents the confidence scores histogram, that is, each bar presents the portion of samples for which the model M_t (where t is the iteration number) gives a score that falls in the confidence interval of the bar. Next, the middle column presents the sampling probability of samples in each bin. Using PosProb the sampling probability of a sample x is equal to $M_t(x)$ (the sampling probability of each bin was flattened to simplify the presentation). Lastly, the right column presents the probability that a sample sampled according to the fitting distribution (appearing in the middle column) would belong to each confidence score interval. In addition, each bar in the left column presents the portion of samples that are negative and positive in each confidence interval.

D Detailed results

In this section more detailed results can be found, specifying all metrics for all imbalance ratios, algorithms and datasets. In addition, several tables omitted from the main article due to space considerations can be found here.

			Balanced Accuracy			AUC			Positives Ratio		
			InvProp	PosProb	Uniform	InvProp	PosProb	Uniform	InvProp	PosProb	Uniform
2.0	ALPS	AG_news	0.9207	0.9205	0.9231	0.9678	0.9675	0.9679	0.4188	0.4061	0.4201
		DBpedia	**0.9877**	0.9865	0.9836	0.9983	0.9983	0.9981	**0.4181**	0.3996	0.3953
		Pubmed	0.8651	0.8672	0.8631	0.9305	0.9322	0.9309	0.4554	0.4550	0.4510
		SST-2	0.8221	0.8489	0.8475	0.9120	0.9209	**0.9240**	0.4566	0.4548	0.4558
	BADGE	AG_news	**0.9280**	0.9270	0.9239	0.9704	0.9698	0.9696	**0.4675**	0.4562	**0.4682**
		DBpedia	0.9926	0.9930	0.9932	0.9983	**0.9990**	**0.9990**	0.4297	**0.4556**	0.4485
		Pubmed	0.8604	0.8572	0.8691	0.9244	0.9278	0.9333	**0.4702**	0.4433	**0.4687**
		SST-2	0.8151	0.8088	**0.8360**	0.9054	0.8968	0.9111	0.4684	0.4700	**0.4796**
	DAL	AG_news	**0.9218**	0.5934	**0.9195**	**0.9672**	0.9448	**0.9670**	0.4651	0.0923	**0.4826**
		DBpedia	**0.9928**	0.8347	0.9860	**0.9987**	0.9952	**0.9985**	0.4565	0.0821	**0.4741**
		Pubmed	**0.8661**	0.5502	0.8596	**0.9303**	0.8853	**0.9280**	0.4578	0.1232	**0.4718**
		SST-2	0.7991	0.5478	**0.8275**	0.8993	0.8490	**0.9123**	**0.4632**	0.1380	**0.4748**
	Random	AG_news	**0.9240**	0.9192	0.9206	**0.9697**	0.9653	0.9674	0.4622	0.3221	**0.4755**
		DBpedia	**0.9937**	0.9858	0.9884	**0.9991**	0.9983	0.9984	0.4308	0.2364	**0.4765**
		Pubmed	**0.8663**	0.8581	0.8637	**0.9325**	0.9272	0.9295	0.4604	0.3588	**0.4738**
		SST-2	0.8043	0.8181	0.8292	0.8988	0.9098	0.9090	**0.4728**	0.4338	**0.4798**
10.0	ALPS	AG_news	0.8442	0.8496	0.8599	0.9526	0.9517	0.9535	0.1120	0.1126	0.1067
		DBpedia	0.9805	0.9792	0.9738	0.9973	0.9969	0.9968	0.0929	0.0929	0.0840
		Pubmed	0.7404	0.7444	0.7574	0.9041	0.8990	0.9070	0.1288	0.1264	0.1234
		SST-2	0.6813	0.6570	0.6663	0.8483	**0.8697**	0.8549	0.1098	**0.1214**	0.1114
	BADGE	AG_news	**0.8738**	0.8632	0.8608	**0.9599**	0.9596	0.9566	**0.4054**	0.3894	0.3169
		DBpedia	0.9895	0.9888	0.9882	0.9980	0.9984	0.9981	0.3919	**0.4164**	0.2928
		Pubmed	0.7406	0.7231	0.7255	0.8999	**0.9066**	**0.9079**	**0.3634**	0.3282	0.2613
		SST-2	0.7165	0.6994	0.6736	**0.8985**	0.8851	0.8768	**0.2982**	0.2480	0.2026
	DAL	AG_news	**0.8789**	0.8441	0.8467	**0.9569**	0.9424	0.9461	**0.4722**	0.2629	0.0927
		DBpedia	**0.9900**	0.9406	0.9778	**0.9975**	0.9960	0.9969	**0.4571**	0.1625	0.0841
		Pubmed	0.7516	0.7443	0.7398	**0.8923**	0.8642	**0.8979**	**0.4429**	0.3992	0.0966
		SST-2	**0.7230**	**0.7405**	0.6568	**0.8849**	**0.8911**	0.8229	0.3950	**0.4438**	0.0962
	Random	AG_news	**0.8663**	**0.8570**	0.8394	**0.9589**	**0.9561**	0.9513	**0.4047**	0.3591	0.0966
		DBpedia	**0.9883**	**0.9883**	0.9843	0.9982	0.9984	0.9980	0.3980	**0.4545**	0.0999
		Pubmed	0.7416	0.7349	0.7430	**0.9079**	0.9024	0.8998	**0.3263**	0.2655	0.0994
		SST-2	0.7039	0.6985	0.7052	**0.8898**	**0.8874**	0.8714	**0.2856**	0.1914	0.0966
20.0	ALPS	AG_news	0.7779	0.7703	0.7618	0.9312	0.9322	0.9272	0.0624	0.0587	0.0571
		DBpedia	0.9585	**0.9725**	0.9641	0.9959	**0.9970**	0.9963	0.0521	0.0479	0.0461
		Pubmed	0.6772	0.6761	0.6737	0.8743	0.8655	0.8647	0.0701	0.0679	0.0680
		SST-2	**0.6110**	**0.6135**	0.5573	**0.8215**	0.8114	0.7834	**0.0666**	0.0596	0.0474
	BADGE	AG_news	**0.8264**	**0.8250**	0.8005	0.9460	0.9475	0.9481	**0.4018**	0.3443	0.2784
		DBpedia	0.9869	0.9871	0.9556	0.9974	0.9980	0.9981	**0.3976**	**0.3743**	0.2756
		Pubmed	0.6509	**0.6724**	0.6598	0.8678	**0.8783**	0.8797	**0.2982**	0.2501	0.1934
		SST-2	0.6411	0.6206	**0.6536**	0.8567	0.8659	0.8561	**0.2432**	0.1722	0.1602
	DAL	AG_news	**0.8199**	**0.8264**	0.7709	**0.9492**	0.9391	0.9241	**0.4219**	**0.4131**	0.0409
		DBpedia	**0.9880**	**0.9877**	0.9479	**0.9968**	0.9959	0.9947	**0.4495**	0.2795	0.0321
		Pubmed	0.6778	0.6616	0.6587	0.8701	0.8606	0.8631	0.3710	**0.4086**	0.0458
		SST-2	0.6172	**0.6761**	0.5661	**0.8683**	**0.8450**	0.7689	0.3114	**0.3808**	0.0396
	Random	AG_news	**0.8277**	0.8103	0.7906	**0.9534**	0.9466	0.9372	**0.3671**	0.2809	0.0534
		DBpedia	**0.9870**	**0.9858**	0.9716	**0.9977**	**0.9982**	0.9968	0.3679	**0.4224**	0.0475
		Pubmed	0.6781	0.6815	0.6695	0.8807	0.8728	0.8719	**0.2726**	0.1845	0.0498
		SST-2	0.6219	**0.6803**	0.6168	**0.8691**	**0.8632**	0.8114	**0.2012**	0.1220	0.0506
50.0	ALPS	AG_news	**0.7219**	0.6911	0.6827	**0.9110**	0.8990	0.8921	**0.0483**	0.0297	0.0301
		DBpedia	0.9376	0.9344	**0.9587**	0.9951	0.9952	**0.9966**	0.0261	0.0260	0.0259
		Pubmed	**0.6007**	0.5840	0.5971	0.8122	0.8043	0.8104	**0.0368**	0.0291	0.0283
		SST-2	**0.5492**	**0.5531**	0.5244	0.7534	0.7546	0.7325	**0.0300**	**0.0256**	0.0202
	BADGE	AG_news	0.7421	**0.7586**	0.7334	0.9216	0.9295	**0.9346**	**0.3337**	0.2846	0.2031
		DBpedia	0.9838	0.9719	0.9669	0.9972	0.9978	0.9974	**0.3627**	**0.3525**	0.2288
		Pubmed	0.5911	0.6042	0.6011	0.8266	0.8136	0.8156	**0.2162**	0.1651	0.1218
		SST-2	0.6045	0.6221	0.6066	0.8343	0.8356	0.8261	**0.1598**	0.1282	0.0990
	DAL	AG_news	**0.7279**	**0.7340**	0.6588	**0.9269**	**0.9253**	0.8899	**0.3115**	**0.3261**	0.0133
		DBpedia	0.9493	**0.9837**	0.9085	**0.9968**	0.9960	0.9937	0.4159	**0.4532**	0.0072
		Pubmed	0.5927	0.5915	0.5880	**0.8393**	**0.8254**	0.7995	0.2218	**0.2378**	0.0180
		SST-2	0.5635	**0.5668**	0.5342	**0.8166**	**0.8328**	0.7231	**0.1730**	**0.1864**	0.0176
	Random	AG_news	**0.7454**	**0.7419**	0.6796	**0.9252**	**0.9234**	0.8885	**0.2850**	0.1677	0.0220
		DBpedia	**0.9830**	0.9797	0.9188	**0.9974**	**0.9980**	0.9929	**0.3481**	0.3145	0.0141
		Pubmed	0.5991	**0.6117**	0.5830	**0.8401**	0.8266	0.8175	**0.1907**	0.0918	0.0210
		SST-2	**0.6109**	**0.5834**	0.5395	**0.8433**	0.8010	0.7665	**0.1410**	0.0498	0.0202
100.0	ALPS	AG_news	0.6592	0.6491	0.6478	0.8791	0.8755	0.8645	**0.0416**	0.0165	0.0152
		DBpedia	**0.9010**	0.8697	0.8831	0.9908	0.9894	0.9856	**0.0211**	0.0099	0.0119
		Pubmed	0.5657	0.5749	0.5698	0.7831	0.7838	0.7742	**0.0272**	0.0161	0.0153
		SST-2	0.5233	0.5126	0.5090	0.7076	0.6989	0.6845	**0.0166**	0.0110	0.0084
	BADGE	AG_news	0.6915	0.6998	0.7036	0.8945	0.9015	0.8897	**0.2806**	0.2180	0.1535
		DBpedia	0.9792	0.9624	0.9778	0.9974	0.9970	0.9972	**0.3399**	0.3012	0.1864
		Pubmed	0.5821	0.5820	**0.5904**	0.7747	0.7866	0.7834	**0.1718**	0.1146	0.0795
		SST-2	0.5585	0.5652	0.5469	0.7718	0.7577	0.7716	**0.0864**	**0.0726**	0.0554
	DAL	AG_news	0.6817	0.6708	0.6684	**0.9016**	0.8842	0.8881	**0.2203**	0.1967	0.0081
		DBpedia	**0.9636**	**0.9475**	0.8369	**0.9960**	**0.9953**	0.9850	**0.4071**	0.3835	0.0039
		Pubmed	**0.5780**	0.5736	0.5618	0.7794	0.7606	0.7784	**0.1498**	0.1390	0.0086
		SST-2	**0.5407**	**0.5644**	0.5150	**0.7515**	**0.7675**	0.6944	0.0782	**0.1058**	0.0082
	Random	AG_news	**0.6997**	**0.7002**	0.6459	**0.8978**	0.8961	0.8789	**0.2512**	0.1090	0.0092
		DBpedia	**0.9785**	0.9741	0.8914	**0.9977**	0.9969	0.9898	**0.3161**	0.2620	0.0060
		Pubmed	**0.5810**	**0.5847**	0.5579	0.7827	0.7818	0.7922	**0.1454**	0.0621	0.0103
		SST-2	**0.5612**	0.5401	0.5266	**0.7954**	0.7467	0.7219	**0.0866**	0.0326	0.0110

Table 8: Results are averaged over 10 seeds and generated Binary-Imbalanced datasets of each described dataset. For each row and metric scores that are statistically significant (as described in Section 5.4) are in bold font (more than one score can be bolded in case of a tie at first place).

			Balanced Accuracy			AUC			Positives Ratio		
			InvProp	PosProb	Uniform	InvProp	PosProb	Uniform	InvProp	PosProb	Uniform
200.0	ALPS	AG_news	**0.8671**	0.8539	0.8461	**0.6360**	0.6051	0.5940	**0.0513**	0.0131	0.0075
		DBpedia	0.9892	0.9904	0.9872	**0.8838**	0.8586	0.8336	**0.0233**	0.0061	0.0056
		Pubmed	0.7495	0.7627	0.7513	**0.5573**	0.5431	0.5423	**0.0219**	0.0083	0.0063
		SST-2	0.6489	**0.6793**	**0.6923**	0.5056	0.5068	0.5074	0.0058	0.0048	0.0044
	BADGE	AG_news	0.8592	0.8703	**0.8817**	0.6598	0.6520	0.6633	**0.2278**	0.1744	0.1237
		DBpedia	0.9954	**0.9971**	**0.9972**	0.9716	0.9595	0.9739	**0.2937**	**0.2688**	0.1609
		Pubmed	0.7287	0.7449	0.7486	0.5478	**0.5672**	**0.5685**	**0.1050**	0.0792	0.0552
		SST-2	**0.7446**	0.6972	0.7236	0.5272	0.5194	0.5143	**0.0382**	0.0284	0.0254
	DAL	AG_news	0.8472	0.8517	0.8566	**0.6402**	**0.6550**	0.6015	**0.1412**	0.1279	0.0032
		DBpedia	**0.9961**	0.9948	0.9916	**0.9424**	**0.9266**	0.8585	**0.3700**	0.3015	0.0019
		Pubmed	0.7379	0.7295	**0.7732**	**0.5652**	0.5616	0.5509	**0.0714**	**0.0722**	0.0051
		SST-2	0.6867	0.6705	0.6831	0.5207	0.5146	0.5128	0.0216	**0.0388**	0.0034
	Random	AG_news	**0.8790**	0.8612	0.8590	**0.6696**	0.6500	0.6191	**0.2024**	0.0670	0.0046
		DBpedia	0.9959	**0.9972**	0.9919	**0.9702**	**0.9717**	0.8775	**0.2879**	0.1989	0.0041
		Pubmed	0.7557	0.7680	0.7738	**0.5677**	0.5581	0.5524	**0.0936**	0.0297	0.0044
		SST-2	0.7045	0.7004	0.6914	**0.5222**	0.5063	0.5136	**0.0340**	0.0110	0.0052
1000.0	ALPS	AG_news	0.7950	0.8124	0.8108	0.5527	0.5461	0.5609	**0.0222**	0.0031	0.0018
		DBpedia	0.9885	0.9890	0.9854	**0.8916**	0.8597	0.8306	**0.0231**	0.0025	0.0012
		Pubmed	0.6941	**0.7234**	**0.7312**	0.5215	**0.5326**	0.5277	**0.0068**	0.0019	0.0011
		SST-2	0.6310	0.6521	**0.6738**	0.5018	0.5019	0.5042	0.0006	0.0006	0.0010
	BADGE	AG_news	0.7612	**0.7979**	**0.7877**	0.5643	0.5771	0.5742	**0.0748**	0.0576	0.0383
		DBpedia	0.9910	0.9929	**0.9941**	0.9262	0.9401	**0.9487**	**0.2631**	0.1872	0.1037
		Pubmed	0.6678	**0.6970**	**0.7165**	0.5323	0.5332	0.5304	**0.0365**	0.0275	0.0160
		SST-2	0.6423	0.6497	0.6540	0.5013	0.5031	0.5029	**0.0036**	0.0026	0.0014
	DAL	AG_news	0.7979	0.7516	**0.8381**	0.5456	0.5533	**0.5879**	**0.0188**	**0.0228**	0.0001
		DBpedia	**0.9946**	0.9884	0.9882	**0.9290**	**0.9363**	0.8471	**0.0927**	**0.0932**	0.0001
		Pubmed	0.7092	0.6467	**0.7573**	0.5199	0.5231	**0.5446**	0.0087	**0.0117**	0.0007
		SST-2	**0.6845**	0.6248	**0.6831**	0.5028	0.5028	0.5028	0.0020	**0.0042**	0.0014
	Random	AG_news	0.8192	0.8252	**0.8473**	0.5768	0.5354	**0.6031**	**0.0639**	0.0109	0.0006
		DBpedia	0.9932	0.9939	0.9920	**0.9448**	**0.9386**	0.8572	**0.2291**	0.0743	0.0004
		Pubmed	0.7080	0.7274	**0.7622**	**0.5366**	0.5250	**0.5396**	**0.0346**	0.0067	0.0013
		SST-2	0.6948	0.6922	0.6889	0.5092	0.5030	0.5053	**0.0058**	0.0012	0.0012

Table 9: Results given an extreme imbalance (i.e., larger than 100) are averaged over 10 seeds and generated Binary-Imbalanced datasets of each described dataset. For each row and metric scores that are statistically significant (as described in Section 5.4) are in bold font (more than one score can be bolded in case of a tie at first place).

		AUC		CI	
Dataset	Seed Size	InvProp	Uniform	InvProp	Uniform
AG_news	0	0.902463	0.889304	0.029820	0.033056
	5	0.921579	0.934612	0.020385	0.014810
DBpedia	0	0.988120	0.978996	0.018172	0.037597
	5	0.997170	0.997449	0.000994	0.001267
Pubmed	0	0.753074	0.762455	0.028951	0.027222
	5	0.826566	0.815649	0.022644	0.017760
SST-2	0	0.760635	0.735419	0.040608	0.045621
	5	0.834284	0.826100	0.022256	0.030999
MEAN	0	0.851073	0.841544	0.029388	0.035874
	5	0.894900	0.893452	0.016570	0.016209

Table 10: Comparison of initial seed size of 0 vs. 5. Throughout the imbalance ratio is 50, and the active learning algorithm is BADGE . We list per dataset the average, taken over 10 training jobs over each of the different positive classes for the dataset (see Section 5.1). The average is for the AUC metric, and the columns describe the weighting scheme applied over the algorithm, as well as whether they describe the AUC metric itself or its Confidence interval (CI). The CI is computed as the standard deviation times 1.96, in order to reflect a 95% CI, under the normal distribution assumption. The bottom entries are the mean over all datasets for seed size of 0 and 5.

AL Algorithm	Number of Bins 2	5	10	15	20
			AUC		
ALPS	0.873414	0.881937	0.879144	0.872674	0.883288
BADGE	0.899262	0.900235	0.894534	0.898439	0.893886
DAL	0.890652	0.896815	0.901556	0.917760	0.905428
Random	0.895781	0.911760	0.897786	0.903739	0.904134
			Balanced Accuracy		
ALPS	0.632292	0.670522	0.659186	0.663423	0.676397
BADGE	0.689007	0.700820	0.691520	0.693307	0.702809
DAL	0.701066	0.701316	0.681660	0.688912	0.686651
Random	0.706186	0.706660	0.699700	0.708197	0.709099

Table 11: Comparison of AUC and Balanced Accuracy of InvProp-weighting with a different number of bins when applied to each algorithm, on the AG-news dataset with an imbalance ratio of 100.

Sampler	P-value per Number of Bins $p(2)$	$p(5)$	$p(15)$	$p(20)$
			AUC	
ALPS	0.593517	0.757289	0.547902	0.640220
BADGE	0.687448	0.614450	0.732451	0.962275
DAL	0.328400	0.609809	0.082592	0.623983
Random	0.884615	0.084827	0.575973	0.466066
			Balanced Accuracy	
ALPS	0.085517	0.416055	0.734206	0.250066
BADGE	0.729087	0.298675	0.858846	0.214114
DAL	0.127080	0.071157	0.617855	0.673392
Random	0.363981	0.342589	0.213729	0.230822

Table 12: P-values for AUC and Balanced Accuracy values, corresponding to Table 11, comparing different numbers of bins with our choice of 10 bins throughout the paper.

Dataset	Balanced Accuracy InvProp	Uniform	AUC InvProp	Uniform	Positives Ratio InvProp	Uniform
agnews	0.734325	0.688625	0.921175	0.901275	0.244625	0.067125
dbpedia	0.963425	0.938225	0.996625	0.995150	0.288200	0.069000
pubmed	0.595900	0.592300	0.829550	0.810750	0.166375	0.047275
sst2	0.582025	0.551175	0.811900	0.762050	0.125950	0.039250