MixBoost: Synthetic Oversampling using Boosted Mixup for Handling Extreme Imbalance

Anonymous Author(s)*

ABSTRACT

Training classification models on datasets where the class distribution is heavily skewed is a challenging problem. Such imbalanced datasets are common in real-world situations such as fraud detection, medical diagnosis and customer churn prediction. We propose a data augmentation method, MixBoost (Synthetic Iterative Oversampling using Mixup). MixBoost intelligently selects and then combines instances from the majority and minority classes to generate synthetic hybrid instances that have elements of both classes. We evaluate MixBoost on 20 benchmark datasets and show that it outperforms existing approaches. We also provide ablation studies to highlight the impact of the different components of our framework.

KEYWORDS

datasets, neural networks, sampling, minority sampling, data augmentation, class imbalance

ACM Reference Format:
Anonymous Author(s). 2018. MixBoost: Synthetic Oversampling using Boosted Mixup for Handling Extreme Imbalance. In *WSDM '20: Web Search and Data Mining, February 03–07, 2020, Houston, Texas, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Consider the problem of training a binary (two-class) supervised classification model on a dataset where the instances of one class far outnumber those of the other class. Figure 1 shows an example of one such imbalanced dataset (the XYZ dataset [???]) and the decision boundary of a Support Vector Machine (SVM) [???] trained on this dataset. The trained SVM is biased towards the class having a large number of instances (the majority class) and performs poorly on the class having a very small number of instances (the minority class). We will describe a method for augmenting the training data that improves classifier performance on such datasets.

In this paper, we make the following contributions:

- We introduce a data augmentation method, *MixBoost* that generates synthetic hybrid instances to augment an imbalanced dataset (Section 2). *MixBoost* has two innovative components. First, it mixes instances of the minority and majority classes to generates synthetic hybrid instances that have elements of both classes (Section 2.4). Second, it intelligently selects the instances for mixing using an entropy weighted technique (Section 2.3).
- We show that *MixBoost* outperforms state-of-the-art methods in synthetic minority oversampling on highly imbalanced benchmark datasets for different levels of class imbalance (Section 4).

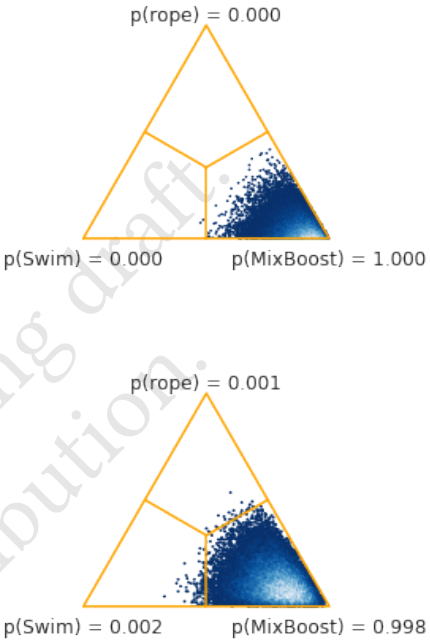

Figure 1: ...

Several real-world problems involve working with and learning from imbalanced datasets. Examples of such problems include fault detection [24], disease classification [18], software failures [3], customer churn prediction [4], oil spill detection [16] and protein sequence detection [2]. In these problems, instances of one class (the majority class) far out-number the instances of the other class (the minority class).

We focus on supervised binary classification problems over such imbalanced datasets. The binary classification problem is particularly challenging when dealing with imbalanced datasets. The most straightforward way to alleviate this problem is by under-sampling the majority class prior to training the classifier. Under-sampling approaches discard instances of the majority class at random to balance the class distribution. However, removing instances of the

WSDM '20, February 03–07, 2020, Houston, Texas, USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

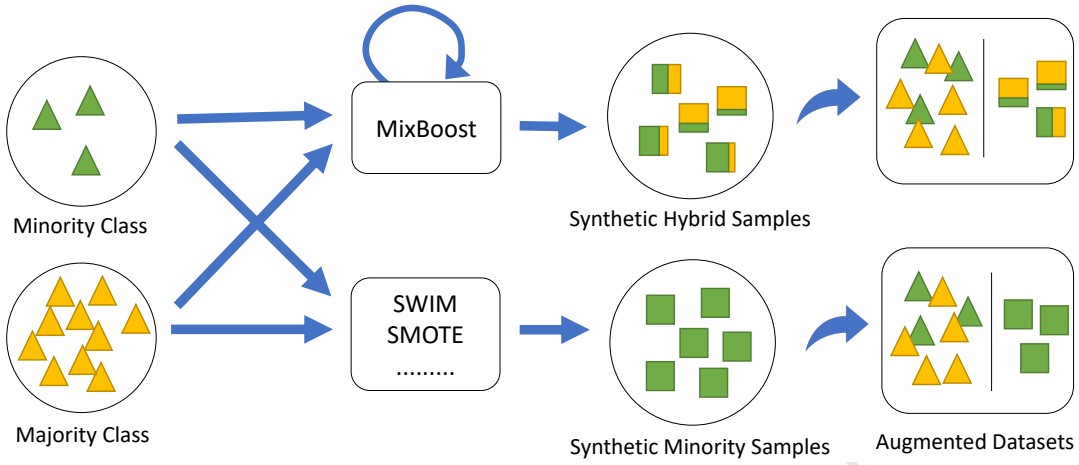

Figure 2: High-level summary of the data-augmentation pipeline. The triangle shape represents the data instances in the non-augmented training dataset. Green is for minority class and yellow for majority class. The square shape represents the synthetic generated instances that are added to the training dataset (augmentation) prior to training. Traditional approaches such as SWIM [19], SMOTE [5] and its variants generate synthetic minority class instances. In contrast, *MixBoost* generates synthetic hybrid instances that interpolate minority and majority class instances.

majority class can lead to a loss of information and subsequently degrade classifier performance. An alternative approach is to balance the class distribution by over-sampling the minority class. Over-sampling approaches duplicate instances of the minority class at random to balance the class distribution. The instances added to the training dataset are duplicates of existing instances and therefore do not add any new information [10, 11]. More sophisticated approaches create synthetic instances to augment the training dataset. SMOTE [5] creates synthetic minority class data by interpolating instances of the minority class. However, SMOTE ignores majority class instances when creating synthetic instances. Variants of SMOTE (Borderline SMOTE[12], ADASYN[15], SMOTEBoost[6]) use the majority class distribution to post-select instances created using the minority class distribution. A limitation of these methods is that using only the minority class instances for data generation restricts the generated synthetic instances to the convex-hull of the minority distribution. To expand the space spanned by generated instances, SWIM [19] inflates the minority class by generating instances along the density contours of the majority class.

However, existing approaches to generate synthetic data are either majority-focused or minority-focused. On one hand, approaches centred on the minority class distribution overlook the majority class, and can degrade classifier performance by generating borderline or overlapping instances. On the other hand, approaches centred on the majority class distribution have a tendency to overfit the training distribution. We propose *MixBoost* (Synthetic Iterative Oversampling using Mixup), a data augmentation method that combines instances from the majority and minority classes to generate synthetic hybrid instances. Each generated synthetic hybrid instance contains elements of a majority class and minority class instance. Further, *MixBoost* uses an information-theoretic measure along with the trained classifier to select the instances to combine. Figure 2 illustrates the essential idea and distinction to related approaches.

MixBoost generates synthetic hybrid instances iteratively. Each iteration consists of a *Boost* step followed by a *Mix* step. In the *Boost* step, *MixBoost* uses the trained classifier to intelligently select pairs of instances from the minority and majority class for mixing. Intuitively, *MixBoost* selects pairs of instances such that one instance in each pair is close to the decision boundary of the trained classifier and the other instance is far from the decision boundary. We use the *entropy* of the predicted class probability vector for the instance as a measure of the distance of the instance to the decision boundary of the classifier. A higher value of *entropy* implies that the classifier is uncertain about the class the instance belongs to and therefore the instance is close to the classifier's decision boundary and vice versa. In the *Mix* step, *MixBoost* mixes the selected instances to create synthetic hybrid instances that are used to augment the training dataset. The classifier is re-trained with the original data augmented with the hybrid instances prior to the next iteration of *MixBoost*. Figure **??** shows a high level overview of *MixBoost*.

We evaluate *MixBoost* against existing state-of-the-art synthetic oversampling approaches, on 20 benchmark datasets (Section 4). In Section 4.2 we conduct ablation studies to evaluate the importance of the *Mix* and *Boost* components of MixBoost .

2 APPROACH

2.1 Definitions

A binary classification task is characterized by a dataset $X^{n \times m} \in R$ and corresponding labels $Y^l \in 0, 1$. The class label 0 corresponds to the majority class, and the class label 1 corresponds to the minority class. n is the number of instances in the dataset. m is the number of features for an instance in the dataset. The training objective is to learn a function, $f(x_i) \rightarrow y_i$, that maps the instances $x_i \in X$

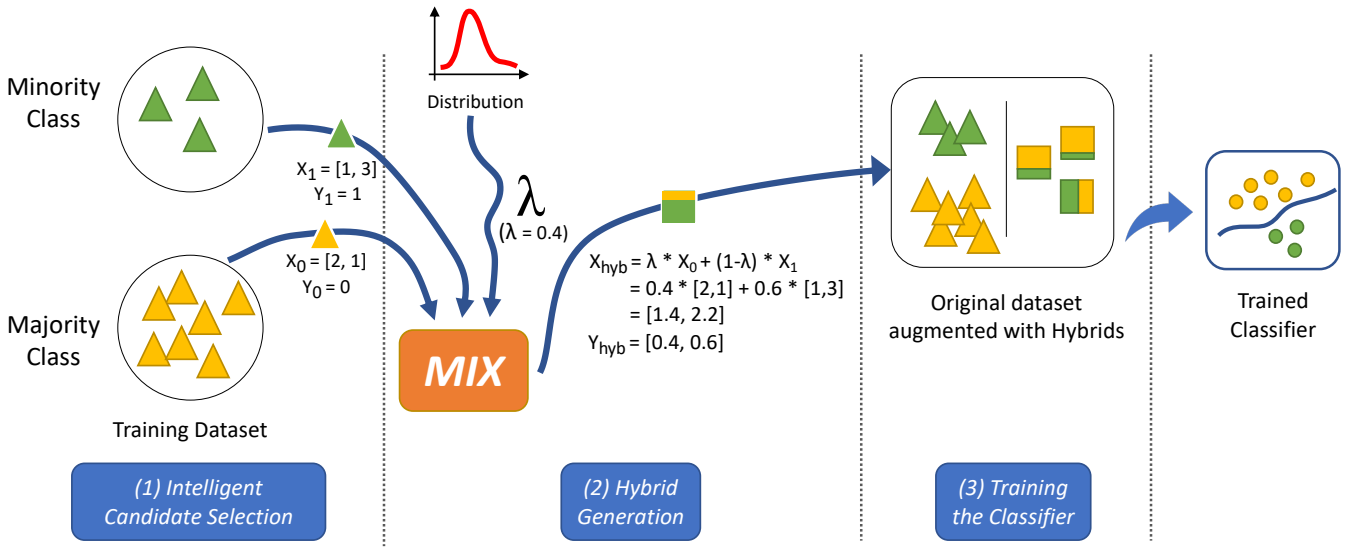

Figure 3: High-level architecture of *MixBoost*. In the (1) candidate selection step (*Boost*), *MixBoost* uses entropy to select instances for mixing from the training dataset. In the (2) hybrid generation step (*Mix*), *MixBoost* mixes the selected instances to create synthetic hybrid instances. λ is the interpolation ratio that controls the extent of overlap between the instances to be mixed. We then augment the training dataset with the generated hybrid instances and train the binary classifier on the augmented dataset.

to their corresponding labels $y_i \in Y$. For imbalanced datasets, the number of instances of the majority class far outnumbers that of the minority class. In general, (X, Y) is split into a training dataset (X_{train}, Y_{train}), and a testing dataset (X_{test}, Y_{test}). n_{train} is the number of instances in the training dataset. We train a binary classification model M on (X_{train}, Y_{train}). M is evaluated on the test set (X_{test}, Y_{test}). We evaluate M using the *g-mean* [17] and ROC-AUC score.

MixBoost uses M, along with (X_{train}, Y_{train}) to create synthetic hybrid training instances. The instances are added to the training dataset to create the augmented dataset $(X^{aug}_{train}, Y^{aug}_{train})$. M is then trained using $(X^{aug}_{train}, Y^{aug}_{train})$. This represents a single iteration of *MixBoost*. *MixBoost* is run for k iterations to expand the augmented dataset. We compute all evaluation scores using the model trained on the final augmented dataset. All evaluation scores are computed on (X_{test}, Y_{test}).

2.2 MixBoost: High-Level Summary

MixBoost generates synthetic hybrid instances by interpolating instances from the majority and minority classes. A single iteration of MixBoost consists of the following steps:

(1) **Model Training:** We train the model M on $(X^{aug}_{train}, Y^{aug}_{train})$. Initially, $(X^{aug}_{train}, Y^{aug}_{train}) = (X_{train}, Y_{train})$.
(2) **Instance Sampling (*Boost*):** We sample instances from the majority and minority classes in (X_{train}, Y_{train}) prior to mixing. We experiment with both random and guided sampling techniques (Section 2.3).
(3) **Hybrid Instance Creation (*Mix*):** We mix the sampled instances to generate synthetic hybrid instances (Section 2.4). (X^{hyb}, Y^{hyb}) denotes the synthetic instances generated in this step.
(4) **Data Augmentation:** We add the synthetic hybrid instances to $(X^{aug}_{train}, Y^{aug}_{train})$. $(X^{aug}_{train}, Y^{aug}_{train}) = (X^{aug}_{train}, Y^{aug}_{train}) + (X^{hyb}, Y^{hyb})$

Figure 3 shows the high-level architecture of *MixBoost*. For the explanation that follows, we consider the problem of generating a single synthetic hybrid instance. The steps to generate several instances follows from the single instance case. Let x_0 and x_1 be instances selected from the majority and minority class, respectively. There are two important considerations. First, how should we select x_0 and x_1 from (X_{train}, Y_{train})? Second, how should we mix x_0 and x_1 to create the synthetic hybrid instance (x^{hyb}, y^{hyb})? We first consider the problem of instance selection.

2.3 MixBoost: Instance Selection (the *Boost* step)

We consider two alternative methods for instance selection.

2.3.1 ***R-Selection.*** We select x_0 and x_1 from (X_{train}, Y_{train}) using Uniform Random Selection or *R-Selection*. The probability of selecting an instance is constant across all instances of the majority and minority classes. Formally, if p^i_0 is the probability of selecting the i^{th} majority class instance and p^j_1 is the probability of selecting the j^{th} minority class instance, then:

$$p^i_0 = 1/n_0 \ \forall i \quad (1)$$

where n_0 is the number of majority class instances in (X_{train}, Y_{train}).

$$p^j_1 = 1/n_1 \ \forall j \quad (2)$$

where n_1 is the number of minority class instances in (X_{train}, Y_{train}). *R-Selection* was proposed in [25].

2.3.2 ***Entropy Weighted (EW) Selection***. *R-Selection* assumes that mixing any two instances to create a hybrid instance is equally useful. However, it is possible that mixing certain instances in the training dataset is more useful than mixing others. Further, the instances selected for mixing should be a function of the trained binary classification model so that the created synthetic instances improve the decision boundary of the model. We propose an Entropy Weighted selection method (*EW-Selection*). *EW-Selection* is based on the intuition that mixing instances closer to the decision boundary of the trained classification model will generate more useful synthetic hybrid instances than mixing instances at random. Formally, if $\vec{y}^i_{pred}$ is the probability distribution over target classes output by M for the i^{th} instance in (X_{train}, Y_{train}), then the Entropy:

$$E^i = E(\vec{y}^i_{pred}) = -\sum y^i_{pred} * log(y^i_{pred}) \tag{3}$$

where the sum is over the elements of the vector $\vec{y}^i_{pred}$. E^i measures the distance of the instance to the decision boundary of the classifier. A high E^i implies that M is uncertain about the correct class for the instance. This case represents an instance that is close to the decision boundary of the classifier. A low E^i implies that M is more certain about the correct class for the instance. This case represents an instance that is far from the decision boundary of the classifier. Let $E^0 = \sum E^i \ \forall x_0$ is the sum of entropy values for majority class instances. Let $E^1 = \sum E^i \ \forall x_1$ is the sum of entropy values for minority class instances. Then,

$$p^i_0 = E^i / E^0 \tag{4}$$

is the probability of selecting a majority class instance for mixing, and

$$p^j_1 = E^j / E^1 \tag{5}$$

is the probability of selecting a minority class instance for mixing. In practice, we find that sampling one instance close to the decision boundary and one far from the boundary leads to the best results. This **low-high** sampling strategy represents a *Mix* between instances where the classifier is certain (low entropy) and where the classifier is uncertain (high entropy).

MixBoost iteratively interleaves entropy-weighted selection (*Boost*) with synthetic hybrid generation (*Mix*). The *Boost* step uses entropy to select instances for mixing from the training dataset (before augmentation). The *Mix* step generates synthetic hybrid instances that are used to augment the training dataset. The augmented training dataset is then used to retrain the classifier. In the next iteration, the retrained classifier is used to generate updated probability vectors (and entropy values) on the original training dataset for the next *Boost* step. We consider next the problem of mixing selected instances.

2.4 MixBoost: Hybrid Creation (the *Mix* step)

Let (x_0, y_0) and (x_1, y_1) be instances sampled (in the *Boost* step) from the majority and minority class respectively. We adopt the mixing strategy introduced in [25].

$$\begin{aligned} x_{hyb} &= \lambda * x_0 + (1 - \lambda) * x_1 \\ y_{hyb} &= \lambda * y_0 + (1 - \lambda) * y_1 \end{aligned} \tag{6}$$

(x_{hyb}, y_{hyb}) is the synthetic hybrid instance. λ is the interpolation ratio. Intuitively, λ controls the extent of overlap between the two instances used to generate the synthetic hybrid instance. In Computer Vision, the augmentation techniques, MIXUP [25] and BC+ [22], are based on linear and non-linear interpolations and they primarily use uniform distributions for sampling λ. BC+ samples λ using the Uniform Distribution $U(0, 1)$ while MIXUP samples λ using the $Beta(\alpha, \alpha)$ distribution. For most tasks, MIXUP uses $\alpha = 1$, which reduces to sampling λ from a Uniform Distribution [20]. In contrast to [25] and [22], we work with skewed data distributions arising from extreme class imbalance. We experiment with sampling λ from linear and non-linear probability density functions in Section 4.2.3.

We also show how easy it is to implement our code from the snippet provided below.

```
from sklearn.datasets import make_classification
from over_sampling import MixBoost

X_train, Y_train = get_data()      # returns the training dataset

interp_dist = lambda: numpy.random.beta(0.5, 0.5)

mb = MixBoost(interp_dist=interp_dist, num_iters=10, \
                    random_state=42)

X_aug, Y_aug = mb.fit_resample(X_train, Y_train)
```

Figure 4: Code snippet that illustrates how to use *MixBoost* in Python. All of our code along with examples is available at ???

3 EXPERIMENTAL SETTINGS

3.0.1 Datasets. To evaluate *MixBoost* against state-of-the-art oversampling methods, we compare performance on binary classification on 20 benchmark datasets (Table 1). To ensure evaluation consistency, we use the same datasets [7] used by [19]. We split the dataset into equal training and testing halves. We randomly down-sample the minority class instances in the training dataset to simulate different levels of extreme imbalance. Specifically, we test at three levels of imbalance, with minority training dataset sizes of 4, 7, and 10. We further skew the data distribution (2 & three minority class instances) to show that *MixBoost* outperforms existing methods when using fewer minority class training instances. Figure 4 illustrates how *MixBoost* can be used to augment a training dataset in Python. For more examples, please refer to ???.

3.0.2 Classification. For all experiments, we use the Naive Bayes, Nearest Neighbour, Decision Tree, Support Vector Machine, and Multi-Layer Perceptron (MLP) binary classifiers. For each dataset, we compare the best performance of each data augmentation method across the various binary classifiers against the performance of *MixBoost* . Since *MixBoost* generates hybrid instances, we use

only the Multi-Layer Perceptron (MLP) classifier to generate evaluation scores for *MixBoost* . We use the *sklearn* package in Python for our experiments; all classifiers are run with default parameters. We use *tensorflow* to implement the MLP. For the MLP, we use Adam optimizer, a learning rate of 0.0001, batch size of 500, and train it for 300 epochs; we use the same hyper-parameters for all datasets. We normalize the data before training and testing.

3.0.3 Data Augmentation. We use each data augmentation method to generate n synthetic instances, where n is the number of instances in the training dataset. For *MixBoost*, we find that generating the n instances over 5 iterations ($0.2n$ for each iteration) leads to the best results. We sample the interpolation ratio λ from a $Beta(0.5, 0.5)$ distribution. We explore the sensitivity of *MixBoost* to these hyper-parameters in Section 4.2.

3.0.4 Evaluation. We compare data augmentation methods on two metrics. First, we use *g-mean* [17], which is the geometric mean of the True Positive Rate (TPR) (for majority class) and the True Negative Rate (TNR) (for minority class).

$$g\text{-}mean = \sqrt{TPR \times TNR} \tag{7}$$

We use *g-mean* because it is both immune to imbalance [17] and provides information about extreme imbalance [19]. Second, for completeness, we also compare methods using the ROC-AUC scores. We repeat all experiments 30 times, and compute the mean and standard deviation of the metrics for each dataset.

We compare *MixBoost* to Random Over-sampling and Under-sampling (ROS, RUS), SMOTE [5], Borderline-SMOTE (B1, B2) [13], SMOTE with Tomek Links [17], ADASYN [15] and SWIM [19]. To make reporting easier, we merge all re-sampling approaches as Alternative Re-sampling Techniques (ALT) and report the best performing combination of data augmentation method and classifier for each dataset. We report the results from SWIM [19] separately since it is the most recent work and it outperforms methods in ALT on most datasets. Finally, *Baseline* represents the case where no data augmentation is used before training the binary classifier.

4 RESULTS

Table 2 shows the results. These results correspond to the case where we down-sample the training dataset to have 4 minority class instances. We see that both *R-Selection* and *EW-Selection* (the two alternative selection strategies in the *Boost* step), outperform existing methods on **18 out of the 20** benchmark datasets. Further, *EW-Selection* outperforms *R-Selection* on **12 out of the 20** datasets. *R-Selection* is better than *EW-Selection* on 6 datasets. This difference between selection strategies for *MixBoost* indicates that selecting instances using entropy leads to the creation of more useful synthetic hybrid instances than selecting instances at random. For completeness, we show in Table 3 the ROC-AUC scores of *MixBoost* and SWIM for the different datasets. For *MixBoost*, we use *EW-Selection* for each dataset. We see that *MixBoost* outperforms SWIM on **19 out of the 20** datasets on the ROC-AUC metric.

Dataset	Features	No. of maj. instances	R4	R7	R10
Abalone 9-18	8	689	1:173	1:99	1:69
Pima Indians	8	500	1:125	1:72	1:50
Diabetes	8	500	1:125	1:72	1:50
Wisconsin	9	444	1:111	1:64	1:456
Wine Q. Red 4	11	1546	1:387	1:221	1:155
Wine Q. White	11	880	1:220	1:126	1:88
Vowel 10	13	898	1:225	1:129	1:90
Vehicle 0	18	641	1:160	1:91	1:64
Vehicle 1	18	624	1:156	1:89	1:62
Vehicle 2	18	622	1:155	1:88	1:62
Vehicle 3	18	627	1:156	1:89	1:62
Ring Norm	20	3736	1:934	1:534	1:374
Waveform	21	600	1:150	1:86	1:60
PC4	37	1280	1:320	1:183	1:128
Piechart	37	644	1:161	1:92	1:65
Pizza Cutter	37	609	1:153	1:87	1:61
Ada Agnostic	48	3430	1:858	1:490	1:343
Forest Cover	54	2970	1:743	1:425	1:297
Spam Base	57	2788	1:697	1:399	1:279
Mfeat Karhunen	64	1800	1:450	1:258	1:180

Table 1: Description of the datasets that we have used in our experiments. To ensure evaluation consistency, we use the same datasets used by [19]. R4, R7, and R10 denote the ratio of class imbalance (minority:majority) after down-sampling the training datasets to have 4, 7, and 10 minority class instances respectively. All datasets can be downloaded from [7]

4.1 Statistical Significance

We use the Bayesian signed test [**?**] to evaluate the results presented in the previous section. The Bayesian signed test is used to compare two classification methods over several datasets. We use the test to compare *MixBoost* to SWIM over all 20 datasets. We compare the methods for training datasets down-sampled to 4, 7 and 10 minority class instances. Using the Bayesian method enables us to ask questions about posterior probabilities, that we cannot answer using null hypothesis tests [19]. Concretely, we wish to ascertain, based on the experiments, what is the probability that *MixBoost* is better than SWIM for data augmentation. We repeat the setup described in [19], where, based on the assumption of the Dirichlet process, the posterior probability for the Bayesian signed test is calculated as a mixture of Diracs deltas centred on the observation.

Figure 5 shows the posterior plots of the Bayesian signed test for the three down-sampled datasets. The posteriors are calculated with the prior parameter of the Dirichlet as $s = 0.5$ and $z0 = 0$ as suggested by [19]. The posterior plots report the samples from the posterior (blue cloud of points), the simplex (the large triangle), and three regions. The region in the bottom left of the triangle indicates the case where it is more probable that SWIM is better than *MixBoost*. The region in the bottom right of the triangle indicates the opposite, that it is more probable that *MixBoost* is better than SWIM. The region in the top of the triangle indicates that it is likely that neither method is better. The closer the points are to the base

Dataset	Baseline	ALT	SWIM[19]	MixBoost	
				R-Selection	EW-Selection
Abalone 9-18	0.481	0.612	0.723	**0.743** ± 0.030	0.735 ± 0.059
Pima Indians	0.276	0.479	0.509	**0.700** ± 0.052	0.597 ± 0.040
Diabetes	0.259	0.509	0.509	**0.701** ± 0.050	0.560 ± 0.036
Wisconsin	0.874	0.956	0.958	0.960 ± 0.021	**0.969** ± 0.080
Wine Q. Red 4	0.224	0.502	0.535	0.714 ± 0.040	**0.815** ± 0.080
Wine Q. White 3v7	0.451	0.572	0.730	0.743 ± 0.060	**0.750** ± 0.050
Vowel 10	0.724	0.738	0.812	0.845 ± 0.043	**0.854** ± 0.066
Vehicle 0	0.534	0.758	0.814	**0.900** ± 0.050	0.850 ± 0.020
Vehicle 1	0.541	0.739	**0.791**	0.700 ± 0.035	0.735 ± 0.030
Vehicle 2	0.450	0.549	0.560	0.880 ± 0.024	**0.638** ± 0.030
Vehicle 3	0.402	0.505	0.569	0.651 ± 0.060	**0.600** ± 0.030
Ring Norm	0.274	**0.933**	0.899	0.550 ± 0.043	0.580 ± 0.027
Waveform	0.301	0.701	0.688	0.812 ± 0.031	**0.844** ± 0.054
PC4	0.572	0.559	0.611	0.720 ± 0.080	**0.737** ± 0.040
PieChart	0.455	0.516	0.576	0.611 ± 0.063	**0.741** ± 0.070
Pizza Cutter	0.468	0.506	0.552	0.725 ± 0.050	**0.678** ± 0.067
Ada Agnostic	0.451	0.445	0.539	**0.690** ± 0.024	0.648 ± 0.033
Forest Cover	0.561	0.554	0.550	0.910 ± 0.048	**0.917** ± 0.020
Spam Base	0.440	0.550	0.685	**0.872** ± 0.030	0.834 ± 0.054
Mfeat Karhunen	0.274	0.933	0.899	0.888 ± 0.068	**0.927** ± 0.050

Table 2: Results (mean and standard deviation) of using *MixBoost* for data augmentation compared to existing over-sampling methods on the *g-mean* metric. The results correspond to the case where we down-sample the training dataset to have 4 minority class instances (R4). Baseline refers to the case where the binary classifier is trained on the original dataset without data augmentation. ALT is the score of the best performing data augmentation strategy (other than SWIM and *MixBoost*). The best score for each dataset is in bold.

Dataset	SWIM	MixBoost
Abalone 9-18	0.790 ±0.083	**0.82** ±0.058
Pima Indians	0.778 ±0.026	**0.80** ±0.015
Diabetes	0.778 ±0.026	**0.80** ±0.015
Wisconsin	0.899 ±0.082	**0.972** ±0.013
Wine Q. Red 4	0.950 ±0.030	**0.971** ±0.028
Wine Q. White 3 vs 7	0.639 ±0.09	**0.783** ±0.053
Vowel 10	0.895 ±0.05	**0.958** ±0.026
Vehicle 0	0.628 ±0.039	**0.896** ±0.07
Vehicle 1	0.592 ±0.06	**0.750** ±0.021
Vehicle 2	0.824 ±0.026	**0.921** ±0.029
Vehicle 3	0.585 ±0.048	**0.664** ±0.027
Ring Norm	0.704 ±0.24	**0.892** ±0.078
Waveform	0.828 ±0.017	**0.869** ±0.02
PC4	0.686 ±0.086	**0.794** ±0.030
PieChart	0.661 ±0.06	**0.743** ±0.079
Pizza Cutter	0.662 ±0.075	**0.735** ±0.039
Ada Agnostic	0.671 ±0.054	**0.798** ±0.024
Forest Cover	0.970 ±0.020	0.970 ±0.020
Spam Base	0.769 ±0.116	**0.835** ±0.071
Mfeat Karhunen	0.980 ±0.01	**0.992** ±0.004

Table 3: Results (mean and standard deviation) of using *MixBoost* for data augmentation compared to SWIM on the ROC-AUC metric. The highest score for each dataset is in bold.

of triangle, the larger is the statistical difference between methods. Figure 5 shows that the probability that *MixBoost* is better than SWIM for data augmentation is high for all three down-sampled training datasets. For all three plots, the point cloud is concentrated in the bottom right triangle, indicating that *MixBoost* is statistically significantly better than SWIM.

4.2 Ablation Studies

We evaluate the importance of the different components of *MixBoost* using ablation studies. To balance the consistency of results and ease of analysis and presentation, we present all results on five datasets. We select these datasets keeping in mind diversity concerning feature dimensionality and the extent of class imbalance. For each study (unless otherwise specified), we augment the training dataset with n synthetic hybrid instances, where n is the number of instances in the training dataset. We sample the interpolation ratio λ from a $Beta(0.5, 0.5)$ distribution. We generate the n synthetic hybrid instances over five iterations, generating $0.2n$ instances in each *MixBoost* iteration. For *MixBoost*, we use *EW-Selection* for each dataset.

4.2.1 Impact of sampling instances iteratively in MixBoost . In this experiment, we evaluate the importance of generating synthetic hybrid instances iteratively. We create a variant of *MixBoost* , called *MixBoost*1*Iteration*, where the n synthetic instances are generated in a single iteration. This single-step generation is in contrast to *MixBoost* , where the n instances are generated over five iterations,

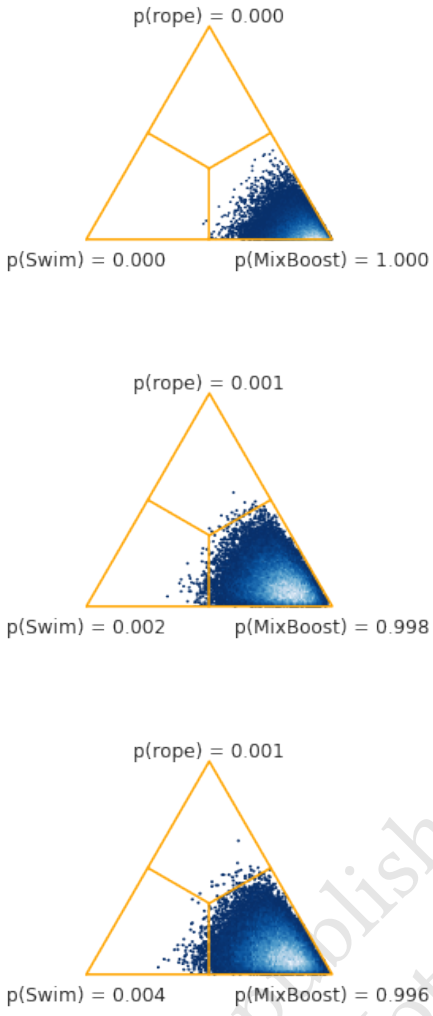

Figure 5: Posteriors for SWIM versus MixBoost (left and right vertex) on the datasets with 4, 7 and 10 minority class instances ((a), (b), and (c)) for the Bayesian sign-rank test. A higher concentration of points on one of the sides of the triangle shows that a given method has a higher probability of being statistically significantly better [19]. The top vertex *rope* indicates the case where neither method is statistically significantly better than the other.

$0.2n$ at each iteration. We augment the training dataset with the generated instances and train the binary classifier on the augmented dataset. Table 4 shows the results. We see that the iterative variant of *MixBoost* outperforms the single-step variant for each dataset. This result illustrates the importance of the iterative element of *MixBoost* .

Dataset	MixBoost1Iteration	MixBoost
Pima Indians	0.510	**0.597**
Waveform	0.841	**0.844**
PC4	0.630	**0.737**
Piechart	0.700	**0.741**
Forest Cover	0.914	**0.917**

Table 4: *g-mean* scores (mean and standard deviation over 30 runs) for the single step and iterative variants of *MixBoost* . For all selected datasets, the iterative variant of *MixBoost* outperforms the single step one.

4.2.2 Impact of reducing the Number of Minority Class Samples. Since *MixBoost* uses an intelligent selection technique for generating synthetic instances, we want to study whether it outperforms existing approaches with a fewer number of minority class training instances. We generate variations of the training dataset with 3 and 2 minority class instances. We then run *MixBoost* on these variations and train a classifier on the augmented dataset. Table 5 shows the results. We can see that *MixBoost* achieves better results than SWIM while using a fewer number of minority class training examples. For instance, for the PC4 dataset, *MixBoost* outperforms SWIM using half the number of minority class instances (2 vs. 4).

Dataset	SWIM	MixBoost		
	4 samples	2 samples	3 samples	4 samples
Pima Indians	0.509	0.582	0.596	**0.597**
Waveform	0.688	0.693	0.727	**0.844**
PC4	0.611	**0.780**	0.770	0.737
PieChart	0.576	0.660	**0.748**	0.741
Forest Cover	0.550	0.412	0.689	**0.917**

Table 5: *g-mean* scores (mean and standard deviation over 30 runs) for data augmentation using MixBoost and SWIM as we reduce the number of minority class instances by down-sampling. For 4 out of the 5 datasets, MixBoost outperforms SWIM using half the number of minority class instances (2 vs 4 respectively).

4.2.3 Selecting different distributions for sampling λ. For sampling the interpolation ratio, λ, we experiment with Linear and Non-Linear Probability Density Functions (PDFs). Table 6 shows the results. On all datasets, using a non-linear PDF ($Beta(\alpha, \alpha)$ with $\alpha = 0.5$) in the *Mix* step leads to better classifier performance.

4.2.4 Impact of Generating Hybrid Instances. Existing approaches to augment training data over-sample by generating synthetic minority class instances. *MixBoost* generates synthetic hybrid instances that have elements of both the minority and majority class. Concretely, in the *Mix* step, we generate instances where the target class vector is in-between the majority class vector $[1, 0]$ and the minority class vector $[0, 1]$. Alternatively, in the *Mix* step, we could have generated synthetic instances where the target class vector corresponded to either the majority or the minority class. To evaluate these alternatives, we create a variant of *MixBoost* in which we assign either the majority (when $\lambda > 0.5$) or the minority class

Dataset	λ-Sampling Distribution	
	Uniform	Beta(0.5,0.5)
Pima Indians	0.355	0.597
Waveform	0.538	0.844
PC4	0.230	0.737
Piechart	0.341	0.741
Forest Cover	0.629	**0.917**

Table 6: *g-mean* scores (mean and standard deviation over 30 runs) when we sample λ (the interpolation ratio for the *Mix* step) from different distributions. For all datasets, sampling λ from the Beta(0.5,0.5) distribution leads to the best performance.

($\lambda <= 0.5$) vector to the generated instance. We label this variant *MixBoostOneHot*. Table 7 shows the results comparing *MixBoost* to *MixBoostOneHot*. We see that generating hybrid instances with in-between target class labels outperforms generating instances with either a majority or a minority class label. We posit that training with hybrid instances that are not explicitly attributed to either class enables the learning of more robust decision boundaries by helping regulate the confidence of the classifier away from the training distribution.

Dataset	MixBoostOneHot	MixBoost
Pima Indians	0.447	**0.597**
Waveform	0.812	**0.844**
PC4	???	**0.737**
Piechart	0.408	**0.741**
Forest Cover	0.894	**0.917**

Table 7: *g-mean* scores (mean and standard deviation over 30 runs) comparing *MixBoost* to a variant where we generate non-hybrid instances (*MixBoostOneHot*) in the *Mix* step. For all considered datasets, generating hybrid instances improves the performance of the binary classifier.

4.2.5 Classifier performance as a function of the number of generated synthetic hybrid instances. *MixBoost* generates synthetic instances iteratively. For each iteration, the classifier is retrained using all instances generated up to this iteration. We want to evaluate the marginal gain in the performance of generating additional instances. We expect that the marginal gain of generating instances should decrease as more instances are generated. Figure 6 validates our expectation. We see that the performance of the classifier plateaus as *MixBoost* adds more synthetic instances to the training dataset. This plateau in performance is important for practical deployment of *MixBoost*. For large datasets, it might not be possible to use *MixBoost* to generate more than twice or thrice the size of the training dataset.

4.2.6 Classifier performance as a function of the number of minority class instances. *MixBoost* mixes an instance of the minority class with one of the majority class to create a synthetic hybrid instance. We expect the quality of the generated synthetic instances to improve as we increase the number of minority class instances in the

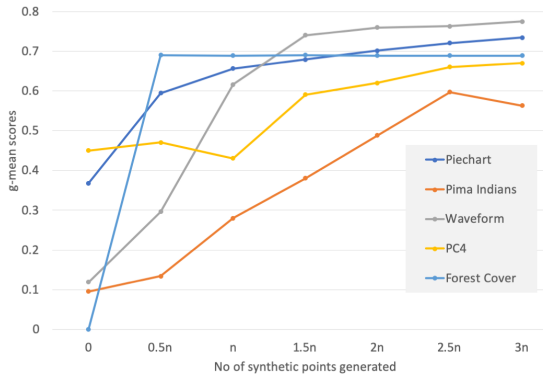

Figure 6: Variation in *g-mean* scores (averaged over 30 runs) for MixBoost as we increase the number of generated synthetic hybrid instances. n represents the total number of training instances in the dataset. The different color lines indicate different datasets. The marginal gain of generating an additional $0.5n$ synthetic instances is initially high and then falls gradually as the number of generated instances increases.

training dataset. The quality of the generated instances is measured by the classifier performance. Figure 7 illustrates that classifier performance increases as we increase the number of minority class instances.

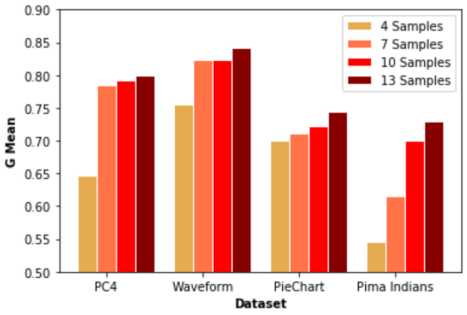

Figure 7: Variation in *g-mean* scores (averaged over 30 runs) for different number of minority class instances. *MixBoost* was used to augment the training dataset in each case. For each dataset, the classifier performance increases with an increase in the number of minority class instances. This increase is more pronounced when we go from 4 to 7 minority samples, after which the classifier performance plateaus.

5 RELATED WORK

In the present work, we focus on supervised binary classification problems over imbalanced datasets.

There are broadly two kinds of approaches for dealing with classification problems for imbalanced datasets. First, sampling based approaches and second, cost based approaches [8, 23]. In the

present work, we focus on sampling approaches. The most straightforward re-sampling strategies are Random under Sampling (RUS), and Random over Sampling (ROS). They balance class distributions by randomly deleting instances of the majority class or randomly duplicating instances of the minority class respectively. However, deleting instances from the training dataset can lead to a loss of information. Further, duplicating instances of the minority class can lead to overfitting. SMOTE (Synthetic Minority Oversampling TEchnique) [5] alleviates these shortcomings.

SMOTE balances the class distribution by generating synthetic data instances based on the k-nearest neighbours of an instance in the minority class. Each synthetic instance is an interpolation of instances in the minority class. SMOTE uses only instances from the minority class to generate synthetic instances. Therefore, the resulting dataset is within the convex hull of the minority class. Further, because it ignores instances of the majority class, SMOTE could increase the overlap between classes. Therefore, if the classes are extremely imbalanced, SMOTE could degrade classifier performance. Extensions of SMOTE [9] focus on a post processing step that removes instances that could degrade the performance of the classifier. These methods incorporate the majority class distribution into the post processing step of the data augmentation process. For instance, Adaptive Synthetic Oversampling (ADASYN) [15], borderline SMOTE [13], and Majority Weighted Minority Oversampling [14] use majority class instances present in the neighbourhood of generated instances for post processing. However, these approaches use the minority class distribution when generating synthetic instances. Therefore, if the number of minority class instances is very small, then the quality of generated instances and subsequently the classifier performance is degraded.

In contrast with euclidean distance in SMOTE-based methods, [1] propose the use of the Mahalanobis Distance for synthetic minority oversampling. However, their generation process does not use information from the majority class. Their work has been extended in SWIM [19]. They use information from the majority class to generate synthetic instances and show the effectiveness of their approach, especially in cases where the minority class is rare. Also, SWIM [19] uses an MD distance based formulation of the majority class, that implicitly assumes a Gaussian distribution, which may not always be true. Tanaka and Aranha [21] introduce a GAN based technique for generating synthetic training samples. However, their technique is unable to consistently outperform state-of-the-art methods. Further, the high compute cost with respect to training time and network parameters limits their practical ability in the domain in contrast to other techniques.

Existing approaches to generate synthetic data are either majority-focused or minority-focused. On one hand, approaches centred on the minority class distribution overlook the majority class, and can degrade classifier performance by generating borderline or overlapping instances. On the other hand, approaches centred on the majority class distribution have a tendency to overfit the training distribution. Our approach, ISWI, uses a probabilistic interpolation strategy that uses information from both the majority class and the minority class to synthesize new instances. ISWI generates synthetic instances by interpolating instances of the minority and majority class. We also introduce an entropy weighted technique for sampling the instances to be interpolated.

Our proposed approach significantly outperforms existing state-of-the-art methods on multiple benchmark imbalanced datasets.

6 CONCLUSION

In the present work, we tackle the problem of binary classification on highly imbalanced datasets. To this end we propose, ISWI, a majority-minority probabilistic interpolation based strategy for generation of synthetic training instances. ISWI generates synthetic training instances by a probabilistic interpolation of instances sampled from the majority and minority classes in the original dataset. We also introduce an entropy-weighted technique to sample the training instances used for interpolation. We extensively evaluate the proposed approach on 20 benchmark imbalanced datasets against existing state-of-the-art techniques and highlight the superiority of ISWI. We also present ablation studies to highlight the impact of each of our contributions and to validate the performance of ISWI in situations of extreme class imbalance.

REFERENCES

[1] Lida Abdi and Sattar Hashemi. 2015. To combat multi-class imbalanced problems by means of over-sampling techniques. *IEEE transactions on Knowledge and Data Engineering* 28, 1 (2015), 238–251.

[2] Ali Al-Shahib, Rainer Breitling, and David Gilbert. 2005. Feature selection and the class imbalance problem in predicting protein function from sequence. *Applied Bioinformatics* 4, 3 (2005), 195–203.

[3] Kwabena Ebo Bennin, Jacky Keung, Passakorn Phannachitta, Akito Monden, and Solomon Mensah. 2017. Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. *IEEE Transactions on Software Engineering* 44, 6 (2017), 534–550.

[4] Jonathan Burez and Dirk Van den Poel. 2009. Handling class imbalance in customer churn prediction. *Expert Systems with Applications* 36, 3 (2009), 4626–4636.

[5] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[6] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. 2003. SMOTEBoost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*. Springer, 107–119.

[7] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[8] Charles Elkan. 2001. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, Vol. 17. Lawrence Erlbaum Associates Ltd, 973–978.

[9] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. 2018. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research* 61 (2018), 863–905.

[10] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2011. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 4 (2011), 463–484.

[11] Salvador García and Francisco Herrera. 2009. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary computation* 17, 3 (2009), 275–306.

[12] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*. Springer, 878–887.

[13] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*. Springer, 878–887.

[14] H He, Y Bai, E Garcia, and S ADASYN Li. 2008. Adaptive synthetic sampling approach for imbalanced learning. IEEE International Joint Conference on Neural Networks, 2008 (IEEE World Congress on Computational Intelligence).

[15] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 1322–1328.

[16] Miroslav Kubat, Stan Matwin, et al. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, Vol. 97. Nashville, USA, 179–186.

[17] Miroslav Kubat, Stan Matwin, et al. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, Vol. 97. Nashville, USA, 179–186.
[18] S Miri Rostami and M Ahmadzadeh. 2018. Extracting predictor variables to construct breast cancer survivability model with class imbalance problem. *Journal of AI and Data Mining* 6, 2 (2018), 263–276.
[19] Shiven Sharma, Colin Bellinger, Bartosz Krawczyk, Osmar Zaiane, and Nathalie Japkowicz. 2018. Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 447–456.
[20] Cecilia Summers and Michael J Dinneen. 2019. Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1262–1270.
[21] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. 2019. Data Augmentation Using GANs. *arXiv preprint arXiv:1904.09135* (2019).
[22] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Between-class learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5486–5494.
[23] Huihui Wang, Yang Gao, Yinghuan Shi, and Hao Wang. 2016. A fast distributed classification algorithm for large-scale imbalanced data. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1251–1256.
[24] Shuo Wang, Leandro L Minku, and Xin Yao. 2014. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering* 27, 5 (2014), 1356–1368.
[25] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).