Learning Objective-Specific Active Learning Strategies with Attentive Neural Processes

Tim Bakker[1](✉), Herke van Hoof[1], and Max Welling[1,2,3,4]

[1] AMLab, University of Amsterdam (UvA)
{t.b.bakker, h.c.vanhoof, m.welling}@uva.nl
[2] Canadian Institute for Advanced Research (CIFAR)
[3] European Lab for Learning and Intelligent Systems (ELLIS)
[4] Microsoft Research (MSR)

Abstract. Pool-based active learning (AL) is a promising technology for increasing data-efficiency of machine learning models. However, surveys show that performance of recent AL methods is very sensitive to the choice of dataset and training setting, making them unsuitable for general application. In order to tackle this problem, the field Learning Active Learning (LAL) suggests to learn the active learning strategy itself, allowing it to adapt to the given setting. In this work, we propose a novel LAL method for classification that exploits symmetry and independence properties of the active learning problem with an Attentive Conditional Neural Process model. Our approach is based on learning from a myopic oracle, which gives our model the ability to adapt to non-standard objectives, such as those that do not equally weight the error on all data points. We experimentally verify that our Neural Process model outperforms a variety of baselines in these settings. Finally, our experiments show that our model exhibits a tendency towards improved stability to changing datasets. However, performance is sensitive to choice of classifier and more work is necessary to reduce the performance the gap with the myopic oracle and to improve scalability. We present our work as a proof-of-concept for LAL on nonstandard objectives and hope our analysis and modelling considerations inspire future LAL work.

Keywords: Active Learning · Deep Learning · Neural Process

1 Introduction

Supervised machine learning models rely on large amounts of representative annotated data and the cost of gathering sufficient data can quickly become prohibitive. Active learning (AL) attempts to mitigate this problem through clever selection of data points to be annotated, thereby reducing total data requirements. To achieve this, AL exploits available information about the dataset and/or supervised task model (e.g. an image classifier) to select data points whose labels are expected to lead to the greatest increase in task model performance. Most classical AL strategies are hand-designed heuristics, based on researcher intuition or theoretical arguments [56]. Recently, much work has been

focused on scaling AL to deep learning (DL) settings, which are even more data-hungry [54]. Such works for instance combine heuristics with representations learned by neural networks [20,59,8,9]), focus specifically on batch acquisition [55,50,7,2,58], or adapt Bayesian Active Learning by Disagreement (BALD) [27,17,61,34,45,33,29]. Despite these developments, it has been observed that modern AL strategies can vary wildly in performance depending on data setting and that there is no single strategy that consistently performs best [3,15,52,54,68,12]. This observation has spurred the development of Learning Active Learning (LAL) methods, which attempt to directly learn an active learning strategy on some data. The goal is to either learn a method that is specifically adapted to the data setting at hand [28,35,24], or to learn a strategy that performs well for various data settings [47,36,40,21]. Such methods have the potential of adapting to additional properties of the task as well, such as nonstandard objectives. A prominent real-life example of such objectives appears in imbalanced data settings, where rare classes are typically more important than their standard contribution to the loss or accuracy suggests. Current active learning surveys generally focus on balanced data settings; few large-scale empirical studies exist for alternative objectives, such as imbalanced data and AL methods designed to work with imbalanced data. In this paper, we propose a novel Learning Active Learning (LAL) method for pool-based active learning. The model learns from a myopic oracle, which gives it the ability to adapt to objectives besides standard classification accuracy. We validate our model in imbalanced data settings, where we show that 1) existing AL methods underperform, and 2) the myopic oracle provides a strong signal for learning. Our contributions are as follows:[5]

1. We show that a wide range of current pool-based AL methods do not outperform uniformly random acquisition on average across multiple deep learning image classification benchmarks. The tested methods generally perform worse on imbalanced data settings than on balanced data settings, suggesting that current AL methods may be under-optimised for the former.
2. We present experiments with a myopic oracle that show large performance gains over standard AL methods on simple benchmarks. We observe that these gains are larger for imbalanced data settings, suggesting the oracle exploits specific highly-informative samples during acquisition.
3. We propose a novel LAL method based on Attentive Conditional Neural Processes that learn from the myopic oracle. The model naturally exploits symmetries and independence properties of the active learning problem. In contrast to many existing LAL methods, it is not restricted to heuristics and requires no additional data and/or feature engineering.

2 Related Work

The field of active learning has a rich history going back decades, with the current taxonomy of methods founded on the extensive survey by [56]. In this work, we

[5] Experiment code can be found at: `https://github.com/Timsey/npal`.

focus on *pool-based* active learning, where a 'pool' of unlabelled data points is available, and the goal is to select one or more of these to label (i.e. 'acquire' the label). Here we focus on some relevant works, and refer to section A of the supplementary material[6], for additional discussion.

The aforementioned survey discusses a number of classical pool-based active learning methods, the most notable among which is Uncertainty Sampling. Here label acquisition is determined by the uncertainty of the classifier. How this uncertainty is measured determines the flavour of Uncertainty Sampling: Entropy selects the points that have maximum predictive entropy, Least Confident acquires the sample on which the task model is least confident in its prediction, and Margin selects the data point with the smallest difference in predicted probability for the first and second most likely class. CoreSet [55] instead take a fully geometric approach to active learning by formulating it as a Core-Set selection problem. Acquisition proceeds through optimising annotated data coverage in some representation space. The authors provide a greedy approximation to their algorithm, called k-Center Greedy, which shows competitive performance while being cheaper to compute. Learning Loss [66] adds a loss prediction module to the base task model, motivated by the idea that difficult-to-classify samples are promising acquisition candidates. This module has the goal of predicting the task model's loss on any given data point and is jointly trained with the task model. Unlabelled samples with the highest predicted loss are then acquired after training.

One potential goal in doing active learning is to select an annotated dataset that represents the true data distribution as well as possible. Based on this idea, Discriminative Active Learning (DAL) [20] learns a classifier (discriminator) to distinguish labelled and unlabelled data based on a representation learned by the task model. Acquisition proceeds by annotating the points that the classifier predicts are most likely to be part of the current unlabelled data pool. Variational Adversarial Active Learning (VAAL) [59] builds on this idea by setting up a two-play mini-max game where a Discriminator network classifies data points as belonging to the labelled or unlabelled set, based on a representation learned by a Variational AutoEncoder (VAE). The VAE is incentivised to fool the discriminator, such that the resulting discriminator probabilities encode similarity between any data point and the currently annotated set. Acquisition then occurs by choosing the least similar points. [8] is a recent Convolutional Graph Neural Network (GCN) method that represents data points as nodes in a graph instead. It too is trained to distinguish labelled and unlabelled datapoints; after training the point with the highest uncertainty according to the GCN is selected for labelling. By representing the full dataset as a graph, this method can encode relevant correlations between data points explicitly. [9] extend this method by using Visual Transformers to learn the graph representation. Although research into active learning methods continues, it has been widely observed that AL strategies performance varies heavily depending on data setting and that there is no

[6] Supplementary material can be found at: `https://github.com/Timsey/npal/blob/main/full_paper.pdf`

single strategy that consistently performs best. Such studies typically focus on balanced data settings [3,15,52,54,68,12].

Active Learning for Imbalanced Data: Compared to the wealth of research on active learning, little work has been done on AL for imbalanced datasets specifically. This further motivates imbalanced data settings as relevant nonstandard objectives for active learning. Existing work in this area typically incorporates explicit class-balancing strategies or additional exploration towards difficult examples. Hybrid Active Learning (HAL) [31] is built on the idea that rare samples may differentiate themselves in feature space. HAL trades off geometry-based exploration (e.g. some average distance to the currently annotated data) with informativeness-based exploitation (e.g. as in Uncertainty Sampling). Class-Balanced Active Learning (CBAL) [5] combines entropy sampling with a regulariser that assigns high values to rare points. This regulariser is the difference between a desired class-histogram (i.e. fully balanced classes) and the sum of softmax values of currently sampled points. This intuitively will have the effect of selecting rare points more often. [10] derives an active learning strategy based on selecting the example with the highest estimated probability of misclassification through Bayes' theorem and various approximate distributions learned by VAE. [1] describe a two-step approach that uses the data's class imbalance profile to switch from classical AL to a class-balancing acquisition function that favours pool points close (in embedding space) to the rarest class in the annotated data. [4] suggests that doing active learning using the variation ratio of a model ensemble may help counteract imbalance in the data.

Learning Active Learning: With the observation that existing AL methods do not consistently perform well across data settings, interest in learning pool-based active learning has risen. The seminal paper by [28] formulates Active Learning By Learning (ALBL) as a multi-armed bandit problem, where the arms are different AL heuristics. The goal is to learn to select the best heuristic for each acquisition round. [24] learns to fine-tune existing AL heuristics using a Bayesian acquisition net trained with the REINFORCE algorithm. [40] instead learn to imitate actions performed by an approximate oracle. Relatedly, [21] reduce the imitation learning goal to a learning-to-rank problem. They meta-train on synthetic data and show this generalises to other datasets. [35] formulates learning active learning as a regression problem. Similarly to our proposed method, they train a model to predict the reduction in generalisation error expected upon adding a label to the dataset. However, their method requires handcrafted global features representing the classification state and annotated dataset as input to their regressor. In contrast, our method implicitly learns the required features from the raw data, allowing for more complex relationships and simplifying engineering choices. Finally, both [47] and [36] perform meta-learning over various binary classification datasets. The former employs a meta-network that encodes dataset and classifier states into parameters for a policy, which is reinforcement learned by the REINFORCE algorithm. The latter employs reinforcement learning with a Deep Q-Network and eschews the meta-network. These methods are either still restricted

to heuristics [28,24], or require gathering additional representative or synthetic datasets for training [53,47,36,40,21], as well as dataset-independent features.

3 A Study on Existing Active Learning Methods

In pool-based active learning, we are given a labeled (classification) dataset $\mathcal{D}_{annot} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=0}^{M}$ of size M, where i indexes the data points, $\boldsymbol{x}_i \in \mathbb{R}^K$ are feature vectors of size K, and $\boldsymbol{y}_i \in \{0,1\}^C$ is a (one-hot) label on C total classes. We are further given an unlabelled dataset $\mathcal{D}_{pool} = \{\boldsymbol{x}_j\}_{j=0}^{N}$ of size N and are tasked with selecting candidates $\boldsymbol{x}_j$ from $\mathcal{D}_{pool}$ to annotate: i.e. select the index j, obtain the label $\boldsymbol{y}_j$, and subsequently add $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ to $\mathcal{D}_{annot}$. The goal of this procedure is to iteratively improve a task model, e.g. a classifier, trained on the annotated data $\mathcal{D}_{annot}$. Improvement is typically measured by some performance metric, e.g. the accuracy on some test dataset $\mathcal{D}_{test}$. Most existing AL methods depend on combinations of heuristics and representation learning for selecting the index j. The implicit expectation is that the selections such heuristics make are also highly performant according to the chosen performance metric. Here we explore whether this assumption holds in modern deep active learning.

Data: To explore the performance of existing heuristic-based AL strategies, we perform active learning on four standard ten-class image classification benchmark datasets: MNIST [11], FashionMNIST [62], SVHN [44], and CIFAR-10 [38]. We use a standard ResNet18 convolutional neural network [25] as the base classifier. We consider three objective settings for each benchmark: Balanced, Imbalanced, and Imbalanced weighted. In imbalanced settings, half the classes are undersampled by a factor 10. Evaluation is performed with a balanced accuracy metric, where instances from undersampled classes are upweighted such that all classes have the same importance. Imbalance weighted additionally takes these weights into account during training. This mimics objectives in typical imbalanced data applications, where rare class instances are often considered more important than common ones [30]. Following [8], we initialise active learning with an annotated dataset $\mathcal{D}_{annot}$ of 1000 data points that follow the specified class ratios; the remaining point also follow these class ratios and are left as the pool dataset $\mathcal{D}_{pool}$. Every acquisition step we batch annotate 1000 points using the specified AL strategy, for a total of ten steps. After each step, we retrain the classifier from scratch. See supplementary B.1 for further implementation details.

AL Strategies: First, we consider the three classical uncertainty sampling strategies [56]: Entropy, Margin and LstConf (least-confident). Second, we include the purely geometric approach of [55]: KCGrdy (K-center greedy). Third, active learning through Learning Loss Module: LLoss [66]. Fourth, Variational Adversarial Active Learning VAAL [59]; a discriminator method based on VAE-learned representations. Fifth, two variations on the same convolutional graph neural network method – UncGCN and CoreGCN [8] – that employ a jointly learned discriminator and graph embedding; unlike VAAL, this approach can

explicitly model inter-datapoint correlations. Sixth, we employ HAL [31] and CBAL [5] as baselines specifically developed for active learning in imbalanced data settings. HAL is further split into HALUni and HALGau, depending on the exploration scheme (uniform or Gaussian). Finally, Random is the uniformly random sampling baseline, corresponding to no active learning.

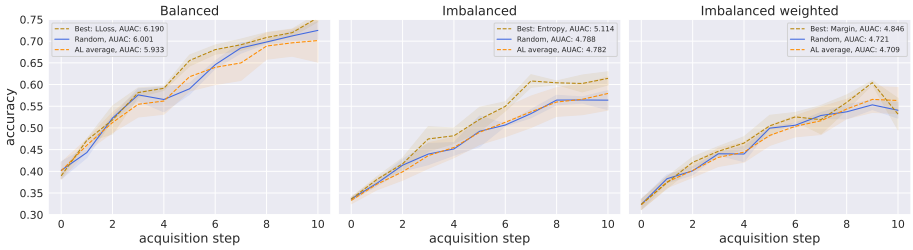

Fig. 1. Random vs. best and average of remaining AL strategies for CIFAR-10 dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels. Shaded region represents standard deviation over three seeds.

Results: In Figure 1 we plot CIFAR-10 test accuracy as a function of acquisition step for Random, the best performing AL method, and the average of all AL methods (excluding Random). AUAC is the Area Under the Acquisition Curve, which is computed as the area under the curves of Figure 1. It measures performance of the whole AL trajectory. We observe that the average active learning strategy does not perform significantly better than Random in any setting. The best performing AL strategy (by AUAC) does outperform Random. These results suggest that AL can be useful, but only if an appropriate strategy is found for the data at hand; a mismatched strategy can lead to performance worse than uniformly random labelling. Note that there is no consistent best performer among the AL methods. This variation in (relative) performance across benchmarks has been previously observed in the literature [3,15,52,54,68,12]. We refer to the supplementary material for implementation details (B.1) and additional results (C.1). We further argue in C.1 that the tested AL methods generally perform worse on the imbalanced objective settings than on the balanced settings, suggesting that current AL methods may be under-optimised for the former.

4 Myopic Oracle Active Learning

Given the results of the previous section, we may wonder if stronger AL strategies can be found. In particular, it would be valuable to develop strategies that perform well out of the box on many different settings. To this end, the field of

Learning Active Learning (LAL) has emerged. The motivating idea is that information about the problem setting should be used for constructing the AL strategy: LAL-methods attempt to do this through learning. What is learned can vary from a choice between existing heuristics [28], to a fine-tuning of such heuristics [24], to a labelling policy that tries to generalise over datasets [47,36,40,21], to the direct improvement to the underlying classifier upon annotating a data point in the given dataset [35]. Ideally, the learned AL strategy should not be constrained to be close to human heuristics, as there is no guarantee that optimal strategies can be represented as such. Additionally, we will only require the availability of a single dataset to train an AL strategy, since finding additional datasets representative of the problem setting at hand is often not feasible in real-world applications. That leaves us with strategies similar to those in e.g. [35], where the AL strategy tries to learn a function mapping the features of an unlabelled datapoint to the expected improvement of the classifier after retraining with that datapoint labelled. Before attempting to train such a strategy, we should quantify whether such a method – if properly learned – actually improves much over existing methods. To this end, we introduce the myopic oracle strategy – denoted ORACLE in the below – which computes the actual classifier improvement on the test data for an unlabelled datapoint $\boldsymbol{x}_j$ in $\mathcal{D}_{pool}$, by treating the corresponding label $\boldsymbol{y}_j$ as known and retraining the classifier with this additional label. This improvement is stored, the classifier is reset, and the process is repeated for every datapoint in $\mathcal{D}_{pool}$. Pseudocode for obtaining improvement scores with the ORACLE is presented in supplementary B.2. ORACLE then selects the datapoint $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ corresponding to the largest classifier improvement and this point is added to the annotated dataset $\mathcal{D}_{annot}$. This oracle uses information that is typically unavailable during the AL process, namely the true labels $\boldsymbol{y}_j$ and the exact classifier improvements on the test set. The oracle is myopic, as it greedily acquires the best datapoint every acquisition step, rather than planning ahead: looking ahead t acquisition steps requires retraining the classifier $\binom{|\mathcal{D}_{pool}|}{t}$ times, which is infeasible.

Classifiers: Even for $t = 1$, the myopic oracle strategy requires retraining the underlying classifier $|\mathcal{D}_{pool}|$ times every acquisition step, which is computationally intractable for neural network classifiers. For this reason, our experiments in this setting use simpler classification models. We run experiments with logistic regression classifiers and provide additional experiments with support vector machine (SVM) classifiers in the supplementary material (C.3). These are both quick-to-train models that have a long history of being used in AL research [56,16,37,68], including within the subfield of LAL [28,35,53,36]. For both classifiers, we employ the default scikit-learn implementations [49], with class-weighting when specified.

Data: These simpler classifiers do not perform well on the image datasets of Section 3. In order to properly study the effects acquisition has on model performance, we instead use simpler datasets. A popular choice in the field of learning active learning [36,53,40] are binary classification datasets from the UCI data repository [13]. We use the 'waveform', 'mushrooms' and 'adult' datasets, since

Table 1. AL strategy AUAC and final-step test accuracy on UCI waveform dataset with logistic regression classifier, 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ORACLE	$\mathbf{9.14} \pm 0.12$	$\mathbf{0.93} \pm 0.01$	$\mathbf{8.84} \pm 0.39$	$\mathbf{0.89} \pm 0.04$	$\mathbf{9.22} \pm 0.18$	$\mathbf{0.93} \pm 0.02$
UNCSAMP	8.67 ± 0.17	0.87 ± 0.01	8.40 ± 0.49	0.85 ± 0.04	8.55 ± 0.33	0.86 ± 0.02
KCGRDY	8.68 ± 0.28	0.87 ± 0.03	8.29 ± 0.49	0.84 ± 0.04	8.58 ± 0.37	0.86 ± 0.03
HALUNI	8.66 ± 0.26	0.87 ± 0.03	8.11 ± 0.55	0.81 ± 0.06	8.45 ± 0.46	0.85 ± 0.05
HALGAU	8.68 ± 0.23	0.87 ± 0.02	8.16 ± 0.54	0.82 ± 0.05	8.48 ± 0.45	0.85 ± 0.04
CBAL	8.67 ± 0.15	0.87 ± 0.02	8.30 ± 0.45	0.84 ± 0.04	8.65 ± 0.34	0.87 ± 0.03
RANDOM	8.65 ± 0.23	0.87 ± 0.02	8.17 ± 0.54	0.82 ± 0.05	8.42 ± 0.48	0.85 ± 0.05
NP	8.69 ± 0.19	0.87 ± 0.02	8.25 ± 0.53	0.83 ± 0.05	8.61 ± 0.33	0.87 ± 0.03

these contain sufficient samples for our experiments post-imbalancing. Data is imbalanced by a factor of ten, as in the previous experiments. In all experiments we initialise the runs with 100 annotated examples and acquire one additional label in each of ten acquisition steps. We set aside 200 datapoints as test data $\mathcal{D}_{test}$ for evaluating the classifiers; oracle scores are also computed on this test data.

AL Strategies: We first compare ORACLE with a logistic regression classifier to the same set of AL strategies we compared to in Section 3. However, we skip the comparisons to LLOSS, VAAL, UNCGCN, and COREGCN, since these all require neural network classifiers as their base. Additionally, the three uncertainty sampling methods ENTROPY, MARGIN, and LSTCONF reduce to the same algorithm for binary classification: we henceforth denote this method as UNCSAMP. Our goal is to work towards a general-purpose AL method that can be trained using only available data. Therefore, we do not include the discussed LAL methods in our baselines, as these methods either adapt existing heuristics or require heavy feature engineering and/or additional datasets to train.

Results: Table 1 compares the performance of the ORACLE to pre-existing AL methods on the waveform dataset for the logistic regression classifier. The NP method will be introduced and discussed in the next section. It is clear that ORACLE dominates all other AL strategies in all settings. Note that AL is only responsible for a small fraction of the total datapoints in the final step here (10 of 110), whereas in the experiments of the previous section, it was responsible for the majority of datapoints (10000 of 11000). As may be observed in the table, such a small number of points is enough to obtain meaningful differences in scores between AL strategies. This indicates that this benchmark contains suffi-

cient variability between strategies to observe meaningful differences in AL quality, making it an appropriate environment for learning active learning. These results suggest that the function represented by ORACLE is a strong active learner that adapts to the given objective. Moreover, we note that the performance gap between ORACLE and RANDOM – and more generally between the various AL strategies – is larger in the imbalanced settings, providing evidence that acquisition choice is more important in these settings; something ORACLE can directly exploit. We refer to the supplementary material for implementation details (B.2) and additional results (C.3). In the next section, we turn our attention to an attempt at learning an approximation to the ORACLE using a Neural Process model.

5 Learning Active Learning with a Neural Process

Our approach will be to learn an approximation to ORACLE, by training a model to predict classifier improvement values for every point in $\mathcal{D}_{pool}$, given a context of annotated datapoints and classifier state. However, we cannot train on the true myopic oracle values, as this requires pool data labels and test data that we do not have access to at training time. Instead, we opt to simulate active learning scenarios by subsampling $\mathcal{D}_{annot}$. For these simulated settings we can compute the improvement values that provide the training signal. Our approach will perform the following procedure at every step of the acquisition process:

1. Simulate many active learning scenarios by subsampling $\mathcal{D}_{annot}$ into N_{sim} pairs of annotated and pool data $\left(\mathcal{S}^{(i)}_{annot},\ \mathcal{S}^{(i)}_{pool}\right)$, with $i \in [1, N_{sim}]$.
2. Use the myopic oracle to compute – for each point in all the $\mathcal{S}^{(i)}_{pool}$ – the classifier improvement observed after retraining with that point and its label to the current dataset $\mathcal{S}^{(i)}_{annot}$.
3. Train a model to predict these improvements from the input $\left(\mathcal{S}^{(i)}_{annot},\ \mathcal{S}^{(i)}_{pool}\right)$.

The challenge is now to design a model and training setup that can generalise strategies learned in the simulated settings to the full test-time AL setting represented by $\mathcal{D}_{annot}$ and $\mathcal{D}_{pool}$. Here we describe our considerations and resulting approach to this challenge. First, the classifier improvements used for training should not be computed using test data, as this data is not available during training. Instead, we compute these scores on a held-out 'reward' dataset $\mathcal{D}_{val}$. In practice, this reward set was used instead of a validation set, so the usual train-val-test split suffices for training our active learner. Second, our problem setup contains permutation symmetries that can be exploited: the (simulated) annotated dataset forms the context that informs the predictions (improvement scores) of our model, but the order of these points does not matter for the prediction: the context representation should be permutation invariant. Additionally, if our model predicts scores for every (simulated) pool datapoint, then these scores should be permutation equivariant: exchanging the index of two pool points should simply exchange the scores. Third, in the myopic setting,

the score of any pool point is independent of any other pool point, so all point points should be treated individually (i.e., not exchange information). This imposes that the model should be invariant to the number of points in $\mathcal{D}_{pool}$. Note that the independence condition is broken in the non-myopic setting, as combinations of pool points can lead to stronger improvements than the individual myopic scores would suggest. The combination of the second and third conditions / inductive biases heavily restrict the choice of model. A natural choice is to use Neural Process (NP) models [19,18,32,14] to learn the approximate ORACLE.

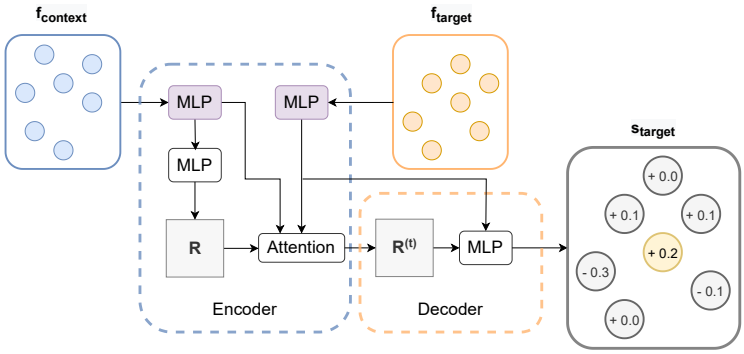

Fig. 2. Computational graph for the Attentive Conditional Neural Process model. The model takes sets of datapoints as input and predicts improvement values for the target points. Context points correspond to annotated data and target points to pool data. All MLPs are applied pointwise. The top two MLPs (in purple) share weights.

The Neural Process: The Neural Process comprises a class of models for meta-learning context-conditional predictors and is a natural choice for our approximator. Given a context $\mathcal{C}$ and target input features $\boldsymbol{f}_\tau$, the Neural Process outputs a distribution $p(\boldsymbol{s}_\tau|\boldsymbol{f}_\tau;\mathcal{C})$ over target predictions $\boldsymbol{s}_\tau$. To apply this model to our problem, we identify the context $\mathcal{C}$ with the information stored in the annotated data and the classifier state, the target input features $\boldsymbol{f}_\tau$ with the features of pool datapoints, and the target predictions $\boldsymbol{s}_\tau$ with the predicted classifier improvements associated to those pool points. We can then train the NP by performing supervised learning – maximising the log likelihood of target improvements $\boldsymbol{y}_\tau$ – on simulated AL scenarios. At test time we apply the trained model with the full $\mathcal{D}_{annot}$ as context and $\mathcal{D}_{pool}$ as target input. In particular, we utilise an Attentive Conditional NP (AttnCNP) [18,32], with cross-attention between the pool and annotated points. The CNP factorises the predictive distribution conditioned on the context set, as

$$p(\boldsymbol{s}_\tau|\boldsymbol{f}_\tau;\mathcal{C}) = \prod_{t=1}^{T} p(s^{(t)}|f_\tau^{(t)};\mathcal{C}), \tag{1}$$

where T is the number of target datapoints. This modelling choice satisfies the independence of pool point predictions. The context $\mathcal{C}$ should be permutation invariant and is typically encoded into a global representation R. The NP is parameterised by a neural network with parameters $\{\theta, \phi\}$ and each factor is typically set to be a Gaussian density [14], as:

$$p_{\theta,\phi}(\boldsymbol{s}_\tau|\boldsymbol{f}_\tau;\mathcal{C}) = p_{\theta,\phi}(\boldsymbol{s}_\tau|\boldsymbol{f}_\tau;R) = \prod_{t=1}^{T} p_{\theta,\phi}(s^{(t)}|f_\tau^{(t)};R) = \prod_{t=1}^{T} \mathcal{N}(s^{(t)};\mu^{(t)},\sigma^{2(t)}), \tag{2}$$

where $R = \text{Enc}_\theta(\mathcal{C})$ encodes the context and $(\mu^{(t)}, \sigma^{2(t)}) = \text{Dec}_\phi(R, \boldsymbol{f}_\tau^{(t)})$ decodes the context encoding and the target features into target predictive parameters. The AttnCNP extends this model by replacing the global representation R with a target-specific representation $R^{(t)}$ through the use of an attention mechanism. In particular, we use the attention mechanism taken from the Image Transformer [48] to perform cross-attention between context and target features, constructing $R^{(t)}$. Here context features $\boldsymbol{f}_\mathcal{C}$ are treated as keys and target features $\boldsymbol{f}_\tau$ as queries. Values are constructed from $\boldsymbol{f}_\mathcal{C}$ by applying a pointwise MLP with 2 hidden layers of size 32 and ReLU activations. Our implementation does not use self-attention on the context or target features, as applying self-attention to the target features violates the independence of the pool point scores. In preliminary experimentation, we found that omitting the attention mechanism – e.g. $R^{(t)} = R$ – resulted in performance drops due to underfitting the target function, as has been observed in the Neural Process literature [32]. A computational graph of our model is presented in Figure 2. This model satisfies the required permutation symmetries while allowing scores of pool points to be given by expressive functions that depend on the context and pool point. In this proof-of-concept study we do not explore the use of uncertainty information for acquisition, rather opting to acquire the datapoint for which $\mu^{(t)}$ – the predicted mean score – is maximal, as $j = \arg\max_{t\in[1,T]} \mu^{(t)}$. We then acquire the pool datapoint with index j, completing a single step in the Active Learning process. The Neural Process is then initialised from scratch, in preparation for the next acquisition step.

Data: The experiments for our Neural Process model (NP) are performed on the datasets described in the previous section. In order to train the NP model, we simulate active learning scenarios by sampling from the existing annotated dataset $\mathcal{D}_{annot}$. We define a set of fractions Q and uniformly sample from these a total of N_{sim} times, leading to a set of annotation fractions $\{q_i\}_{i=1}^{N_{sim}}$. For each value of i, we then assign the corresponding fraction q_i of datapoints from $\mathcal{D}_{annot}$ to a *simulated* annotated dataset $\mathcal{S}_{annot}^{(i)}$; the remaining points are assigned to a *simulated* pool dataset $\mathcal{S}_{pool}^{(i)}$. This procedure results in a set of N_{sim} simulated/sampled active learning problems of various sizes. We then compute oracle scores of all pool points in each of the resulting AL problems $(\mathcal{S}_{annot}^{(i)}, \mathcal{S}_{pool}^{(i)})$. Since we do not have access to test data at train time, the oracle scores are instead computed on the held-out $\mathcal{D}_{val}$. We present pseudocode in Algorithm 1. Experimentally we find that simulating with a variety of fractions in Q improves

Algorithm 1: Training the NP model.

Data: Annotated dataset $\mathcal{D}_{annot}$, Neural Process model NP_θ with parameters θ and fitting function .FIT(.), number of simulations N_{sim}, set of fractions Q, oracle ORACLE, base classifier model C, scoring function SCORE evaluated on $\mathcal{D}_{val}$.
Result: Trained parameters θ^* for NP.
for $i = 1, 2, ..., N_{sim}$ **do**
  $q_i \leftarrow \text{sample}(Q)$; `/* Uniformly sample an 'annotation fraction' */`
  $S^{(i)}_{annot} \leftarrow \varnothing$; `/* Initialise a simulated annotated set */`
  $S^{(i)}_{pool} \leftarrow \mathcal{D}_{annot}$; `/* Initialise a simulated pool set */`
  while $|S^{(i)}_{annot}| < round(q_i \cdot |\mathcal{D}_{annot}|)$ **do**
    Sample index j of datapoints in $\mathcal{D}_{annot}$ uniformly without replacement ;
    $S^{(i)}_{annot} \leftarrow S^{(i)}_{annot} \cup (\boldsymbol{x}_j, \boldsymbol{y}_j)$;
    $S^{(i)}_{pool} \leftarrow S^{(i)}_{pool} \setminus (\boldsymbol{x}_j, \boldsymbol{y}_j)$;
  end
  $V_i \leftarrow \text{ORACLE}\left(S^{(i)}_{annot}, S^{(i)}_{pool}, C, \text{SCORE}\right)$; `/* Obtain improvement scores with Algorithm 2 (supplementary material) */`
end
$\theta^* \leftarrow \text{NP}_\theta.\text{FIT}\left(\{S^{(i)}_{annot}, S^{(i)}_{pool}, V_i\}_{i=1}^{N_{sim}}\right)$; `/* Train the NP on the simulated AL settings */`
return θ^*

generalisation to the target problem over using a fixed single fraction. Our experiments use $Q = \{0.1, 0.2, ..., 0.8, 0.9\}$ and $N_{sim} = 300$. Preliminary experimentation showed no performance increase for larger values of N_{sim}, while using $N_{sim} = 100$ led to slight performance decreases. The held-out dataset $\mathcal{D}_{val}$ consists of the same 100 datapoints for all i.

Results: Table 1 shows the performance of our method – NP – on the UCI waveform dataset with logistic regression classifier. Ignoring ORACLE, the Neural Process ranks best of all active learning methods in AUAC on the Balanced setting, second on Imbalanced weighted, and fourth on Imbalanced. In Figure 3 we show the performance difference between our method and a chosen baseline. Here we choose the average of AL strategies – AL AVERAGE – as the baseline, where we exclude ORACLE, RANDOM, and NP from the average. This choice of baseline allows us to clearly see whether any particular method is expected to improve over a naive application of active learning.

We also show the performance of the best AL strategy – BEST – again excluding ORACLE, RANDOM, and NP from the selection. This represents the relative performance of choosing the best AL strategy post-hoc. We observe that NP performs on par with BEST for Balanced and Imbalanced weighted, and performs similarly to AL AVERAGE for the Imbalanced setting. In all cases, the gap with ORACLE remains large, indicating potential room for improvement. Shaded

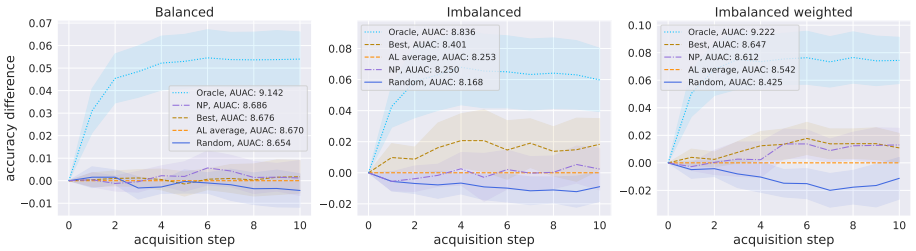

Fig. 3. Relative performance of acquisition strategies for waveform dataset and logistic regression classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). The shaded region represents twice the standard error of the mean over nine seeds.

regions correspond to twice the standard error of the mean, i.e., $2 \cdot \frac{\sigma}{\sqrt{n}}$, where σ is the standard deviation and n the number of runs. Tables 9, 10, and Figures 10, 11 of supplementary C.3 similarly show performance on respectively the mushrooms and adult datasets. NP at least slightly outperforms AL AVERAGE on Imbalanced and Imbalanced weighted for these datasets and in half those cases achieves near-BEST performance. However, NP ranks near the bottom in the Balanced setting here. Interestingly, RANDOM outperforms almost all methods on Balanced, possibly indicating increased difficulty in active learning, although ORACLE does still demonstrate a large performance gap.

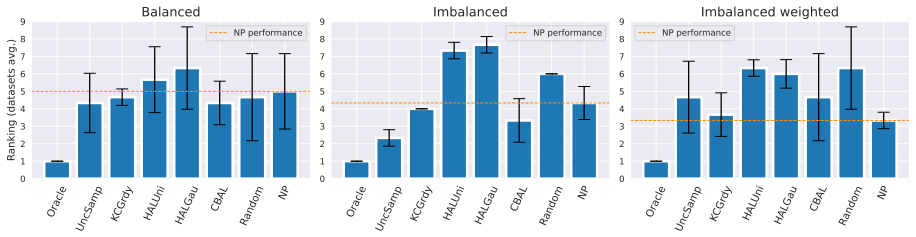

Fig. 4. Relative AUAC rank of AL strategies averaged over the three UCI datasets for logistic regression. The standard deviation of this rank is denoted by the error bars.

Our method is partially motivated by the need for AL algorithms that perform more stably across different data settings. To this end, Figure 4 shows the average AUAC ranking of every AL method across the three UCI datasets. We observe that NP is the best performing AL method on average for the Imbalanced weighted setting and has more middling performance for the other two set-

tings, with Balanced being the worst for our model. Inspecting the ranking standard deviation, we further see that our model achieves a relatively stable ranking across the three datasets in the Imbalanced weighted setting. This stability again degrades for Imbalanced and even further for Balanced. However, note that low standard deviation is only desirable for models with low performance rank, as it otherwise indicates a stable underperformance. These results suggest that NP is better able to exploit information encoded by the ORACLE in imbalanced settings. We present results for the SVM classifier in supplementary C.3. This setting seems more difficult for NP to learn, suggesting that the choice of the underlying classifier is important. Figure 13 corroborates this hypothesis, although it again suggests that NP is specifically more promising for imbalanced settings.

6 Conclusion and discussion

It has been observed in the literature that a wide range of current pool-based active learning methods do not perform better than uniform acquisition on average across standard deep learning benchmarks. We have experimentally verified these results and extended them to imbalanced data settings, which are relevant alternative objectives for many real-world applications. We have explored the validity of using a myopic oracle as a target function for learning active learning (LAL) and have shown its dominating performance on simple active learning tasks. Finally, we have identified symmetry and independence properties of such active learning problems and have modelled these using an Attentive Conditional Neural Process. Unlike existing LAL methods, our model (NP) is not based on existing heuristics, and requires no feature engineering and/or additional datasets to train. Our model generally outperforms the average of the competing AL methods in imbalanced data settings, and occasionally all of them individually. However, future work is needed to evaluate performance on additional datasets, reduce the performance gap with the myopic oracle, and improve scalability. We present our work as a proof-of-concept for LAL on nonstandard objectives – with a focus on imbalanced data settings – and hope our analysis and modelling considerations inspire future LAL work.

Limitations: The primary limitation of our Neural Process approach is scalability. Supervised learning on the myopic oracle requires retraining the base classifier a large number of times during NP training, which is infeasible for large neural network models. Future work may also explore to which degree functions learned on simple classifiers can be transferred to more powerful models. Additionally, the acquisition procedure may be improved through the use of uncertainty information naturally present in the Neural Process. Finally, the Neural Process input may be augmented with additional features – such as the predicted class probabilities of pool points – to potentially improve learning. Preliminary experimentation showed little effect on performance, perhaps due to the large number of raw data features compared to extra features. We leave for exploring the use of extra features for future work.

Ethical Considerations. In recent years, machine learning has had a large impact on society by enabling the development of a variety of new, widely-deployed technologies. Opinions on the value of such technologies vary, but it is clear that they have had both positive and negative impacts. Our research topic of active learning is a promising technology for increasing the efficiency of machine learning model training. Developments in active learning may reduce barrier-to-entry for training and deploying high-performing predictive models, which potentially has both positive and negative downstream consequences. On the positive side, wider access to strong models may increase the adoption of life-saving or simply quality-of-life-improving technologies. Additionally, it may allow relatively less powerful interest groups to not fall behind larger or more powerful institutions in capabilities, thus supporting democratisation of AI. On the negative side, improved active learning has the potential to exacerbate the negative effects of machine learning applications as well. Such exacerbation may happen through widening the aforementioned capability gap between less and more powerful institutions (e.g., by potentially easing model scaling), or through reinforcing existing model biases during training. Additionally, training large-scale models consumes a large amount of energy, potentially worsening the current energy and climate crises. Finally, any machine learning capabilities research potentially exacerbates the future risks of AI misalignment; risks that are worrying to an increasing share of the research community [22,26].

Acknowledgments This work is supported by the 'Efficient Deep Learning' (EDL, `https://efficientdeeplearning.nl`) research programme, which is financed by the Dutch Research Council (NWO) domain Applied and Engineering Sciences (TTW). We are grateful to the Weights&Biases team [6] for providing their experiment tracking software.

References

1. Aggarwal, U., Popescu, A., Hudelot, C.: Active learning for imbalanced datasets. WACV (2020)
2. Ash, J.T., Zhang, C., Krishnamurthy, A., Langford, J., Agarwal, A.: Deep batch active learning by diverse, uncertain gradient lower bounds. ICLR (2020)
3. Baram, Y., El-Yaniv, R., Luz, K.: Online choice of active learning algorithms. JMLR (2004)
4. Beluch, W.H., Genewein, T., Nurnberger, A., Kohler, J.M.: The power of ensembles for active learning in image classification. CVPR (2018)
5. Bengar, J., van de Weijer, J., Fuentes, L., Raducanu, B.: Class-balanced active learning for image classification. WACV (2022)
6. Biewald, L.: Experiment tracking with Weights and Biases. `https://www.wandb.com/` (2020)
7. Bıyık, E., Wang, K., Anari, N., Sadigh, D.: Batch Active Learning Using Determinantal Point Processes. arXiv e-prints (2019)
8. Caramalau, R., Bhattarai, B., Kim, T.K.: Sequential Graph Convolutional Network for Active Learning. CVPR (2021)

9. Caramalau, R., Bhattarai, B., Kim, T.K.: Visual Transformer for Task-aware Active Learning. arXiv e-prints (2021)
10. Choi, J., Moo Yi, K., Kim, J., Choo, J., Kim, B., Chang, J.Y., Gwon, Y., Chang, H.J.: VaB-AL: Incorporating Class Imbalance and Difficulty with Variational Bayes for Active Learning. CVPR (2020)
11. Deng, L.: The MNIST database of handwritten digit images for machine learning research. IEEE-SPM (2012)
12. Desreumaux, L., Lemaire, V.: Learning active learning at the crossroads? evaluation and discussion. arXiv e-prints (2020)
13. Dua, D., Graff, C.: UCI machine learning repository. `http://archive.ics.uci.edu/ml` (2017)
14. Dubois, Y., Gordon, J., Foong, A.Y.: Neural process family. `http://yanndubs.github.io/Neural-Process-Family/` (2020)
15. Ebert, S., Fritz, M., Schiele, B.: RALF: A reinforced active learning formulation for object class recognition. CVPR (2012)
16. Ertekin, S., Huang, J., Bottou, L., Giles, L.: Learning on the border: Active learning in imbalanced data classification. CIKM (2007)
17. Gal, Y., Islam, R., Ghahramani, Z.: Deep bayesian active learning with image data. ICML (2017)
18. Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y.W., Rezende, D., Eslami, S.M.A.: Conditional neural processes. ICML (2018)
19. Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D.J., Eslami, S.M.A., Whye Teh, Y.: Neural Processes. arXiv e-prints (2018)
20. Gissin, D., Shalev-Shwartz, S.: Discriminative Active Learning. arXiv e-prints (2019)
21. Gonsior, J., Thiele, M., Lehner, W.: ImitAL: Learning Active Learning Strategies from Synthetic Data. arXiv e-prints (2021)
22. Grace, K.: What do ML researchers think about AI in 2022? `https://aiimpacts.org/what-do-ml-researchers-think-about-ai-in-2022/`
23. Gómez-Silva, M., Armingol, J., de la Escalera, A.: Balancing people re-identification data for deep parts similarity learning. JIST (2019)
24. Haussmann, M., Hamprecht, F., Kandemir, M.: Deep active learning with adaptive acquisition. IJCAI (2019)
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CVPR (2016)
26. Hendrycs, D.: Statement on AI Risk. `https://www.safe.ai/statement-on-ai-risk`
27. Houlsby, N., Huszár, F., Ghahramani, Z., Lengyel, M.: Bayesian active learning for classification and preference learning. arXiv e-prints (2011)
28. Hsu, W.N., Lin, H.T.: Active learning by learning. AAAI (2015)
29. Jesson, A., Tigas, P., van Amersfoort, J., Kirsch, A., Shalit, U., Gal, Y.: CausalBALD: Deep Bayesian Active Learning of Outcomes to Infer Treatment-Effects from Observational Data. NeurIPS (2021)
30. Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. JBD (2019)
31. Kazerouni, A., Zhao, Q., Xie, J., Tata, S., Najork, M.: Active Learning for Skewed Data Sets. arXiv e-prints (2020)
32. Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., Teh, Y.W.: Attentive neural processes. ICLR (2019)

33. Kirsch, A., Gal, Y.: PowerEvaluationBALD: Efficient Evaluation-Oriented Deep (Bayesian) Active Learning with Stochastic Acquisition Functions. arXiv e-prints (2021)
34. Kirsch, A., Rainforth, T., Gal, Y.: Test Distribution-Aware Active Learning: A Principled Approach Against Distribution Shift and Outliers. arXiv e-prints (2021)
35. Konyushkova, K., Sznitman, R., Fua, P.: Learning Active Learning from Data. NeurIPS (2017)
36. Konyushkova, K., Sznitman, R., Fua, P.: Discovering General-Purpose Active Learning Strategies. arXiv e-prints (2018)
37. Kremer, J., Steenstrup Pedersen, K., Igel, C.: Active learning with support vector machines. DMKD (2014)
38. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
39. Li, T., Cao, P., Yuan, Y., Fan, L., Yang, Y., Feris, R., Indyk, P., Katabi, D.: Targeted supervised contrastive learning for long-tailed recognition. CVPR (2022)
40. Liu, M., Buntine, W., Haffari, G.: Learning how to actively learn: A deep imitation learning approach. ACL (2018)
41. Liu, Z., Ding, H., Zhong, H., Li, W., Dai, J., He, C.: Influence selection for active learning. ICCV (2021)
42. Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., Yu, S.X.: Large-scale long-tailed recognition in an open world. CVPR (2019)
43. Mayer, C., Timofte, R.: Adversarial Sampling for Active Learning. WACV (2018)
44. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. NeurIPS workshop (2011)
45. Nikoloska, I., Simeone, O.: BAMLD: Bayesian Active Meta-Learning by Disagreement. arXiv e-prints (2021)
46. Oh, S., Lee, M.S., Zhang, B.T.: Ensemble learning with active example selection for imbalanced biomedical data classification. IEEE/ACM TCBB (2011)
47. Pang, K., Dong, M., Wu, Y., Hospedales, T.: Meta-Learning Transferable Active Learning Policies by Deep Reinforcement Learning. arXiv e-prints (2018)
48. Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. ICML (2018)
49. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. JMLR (2011)
50. Pinsler, R., Gordon, J., Nalisnick, E.T., Hernández-Lobato, J.M.: Bayesian batch active learning as sparse subset approximation. NeurIPS (2019)
51. Platt, J.C.: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods, p. 61–74. MIT Press (1999)
52. Ramirez-Loaiza, M.E., Sharma, M., Kumar, G., Bilgic, M.: Active learning: An empirical study of common baselines. DMKD (2017)
53. Ravi, S., Larochelle, H.: Meta-learning for batch mode active learning. ICLR workshop (2018)
54. Ren, P., Xiao, Y., Chang, X., Huang, P.Y., Li, Z., Gupta, B.B., Chen, X., Wang, X.: A Survey of Deep Active Learning. arXiv e-prints (2020)
55. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. ICLR (2018)
56. Settles, B.: Active learning literature survey. Tech. rep., University of Wisconsin–Madison (2009)

57. Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., Meng, D.: Meta-weight-net: Learning an explicit mapping for sample weighting. NeurIPS (2019)
58. Shui, C., Zhou, F., Gagn'e, C., Wang, B.: Deep active learning: Unified and principled method for query and training. AISTATS (2020)
59. Sinha, S., Ebrahimi, S., Darrell, T.: Variational Adversarial Active Learning. ICCV (2019)
60. Tran, T., Do, T.T., Reid, I.D., Carneiro, G.: Bayesian generative active deep learning. ICML (2019)
61. Woo, J.O.: BABA: Beta Approximation for Bayesian Active Learning. arXiv e-prints (2021)
62. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv e-prints (2017)
63. Xu, M., Kazantsev, G.: Understanding goal-oriented active learning via influence functions. NeurIPS workshop (2019)
64. Yang, Y., Xu, Z.: Rethinking the value of labels for improving class-imbalanced learning. NeurIPS (2020)
65. Yang, Y., Zha, K., Chen, Y., Wang, H., Katabi, D.: Delving into deep imbalanced regression. ICML (2021)
66. Yoo, D., Kweon, I.S.: Learning Loss for Active Learning. CVPR (2019)
67. Yuille, A., Wei, C., Mellina, C., Yang, F., Sohn, K.: Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. CVPR (2021)
68. Zhan, X., Chan, A.B.: ALdataset: a benchmark for pool-based active learning. arXiv e-prints (2020)
69. Zhang, X., Fang, Z., Wen, Y., Li, Z., Qiao, Y.: Range loss for deep face recognition with long-tailed training data. ICCV (2017)
70. Zhu, J.J., Bento, J.: Generative Adversarial Active Learning. arXiv e-prints (2017)

A Additional related work

Active learning being such a large research field, it is naturally infeasible to refer to all related work. Nonetheless, we include some additional related research here – primarily on balanced non-LAL settings – to give a more comprehensive overview of the literature.

Two more notable classes of active learning strategies discussed in the classical survey [56] are Query-by-Committee (QBC), and Expected Error Reduction (EER). In QBC, multiple task models are trained on the available labelled data, with each model representing a different valid hypothesis. The acquired point(s) are those on which the models have the strongest disagreement in their predictions, by some metric of disagreement, such as such as vote entropy or Kullback-Leibler (KL) divergence. EER seeks to acquire labels for samples that lead to the biggest reduction in generalisation error, typically estimated as the expected error (under the current task model) of the task model on the remaining unlabelled data points. Bayesian Active Learning by Disagreement (BALD) [27] comprises a class of methods that aim to minimise the task model's expected posterior predictive entropy upon acquisition. First adapted to Deep Learning methods by [17], this algorithm has spawned many variations in the literature [61,34,45,33,29]. Another Bayesian approach, motivated by matching the log posterior after acquisition to the full data log posterior, is explored in [50]. A non-Bayesian distribution matching approach [58] incorporates a diversity term that minimises Wasserstein distance between the full data distribution and the acquired data at any given AL step.

Since retraining deep learning models with only a single extra data point is expensive and unlikely to be useful, much DL active learning research has been focused on batch efficiency: selecting batches of data, where individual data points do not strongly overlap in information. This introduces a trade-off between *informativeness* and *diversity* of data selected by AL. Some works explicitly model this trade-off: [7], which makes a principled informativeness-diversity trade-off using k-Determinantal Point Processes (k-DPP). BADGE [2] uses hallucinated gradients as an informativeness proxy, and combines it with a k-DPP sampling procedure that incorporates sample diversity. Generative AL [70,43,60] comprises a class of methods that use GAN-like models to generate informative samples artificially. These may then be annotated directly. These methods suffer from the problem of generating samples that are meaningless to human annotators. As such, unlabelled points from the pool dataset that are closeby to the generated samples according to some metric are often used as alternative acquisition targets. Influence methods [63,41] attempt to use influence functions to estimate changes in the task model that would occur as a result of labelling a particular sample and retraining the task model. This is similar to our method in that influence-based AL can be used to directly estimate model improvements on any arbitrary differentiable utility function. However, these estimations are based on local linearisations of the loss surface, rather than trained on observed model improvements. Moreover, brief experimentation suggested their performance is highly dependent on finding a stable initialisation for the influence estimation.

Imbalanced Machine Learning: There is a large literature on solutions to data imbalance which do not involve active learning. [30] discuss a variety of such methods published on deep learning settings between 2015 and 2018. More recent works cover a wide variety of topics, such as loss function learning for long-tailed data [69], person re-identification [23], transfer learning for open-world settings [42], meta-learning class weights [57], the value of imbalanced labels [64], imbalanced regression [65], semi-supervised learning with minority classes [67], and self-supervised contrastive learning for long-tailed data [39]. Of special interest is [46], which performs a form of active pruning: selecting examples from an existing pool of labelled data to train an ensemble of classifiers. Note that this differs from active learning scenarios in that it requires all data to be already labelled.

B Implementation details

This Supplementary material contains implementation details and additional results for all our experiments. Section B.1 presents ResNet18 experiments, section B.2 presents Myopic Oracle experiments, and section B.3 presents Neural Process experiments.

B.1 ResNet experiments

In this section, we present some additional implementation details for the ResNet experiments of Section 3.

Classifier. We employ a standard ResNet18 [25] implementation for our classifier model, taken from [8]. The classifier is trained on batches of 128 samples for 200 epochs, with a learning rate of 0.1, momentum 0.9, and weight decay 0.0005. The learning rate is decreased by a factor 10 after 160 epochs. Class-weighted training is performed by adding the relative class weights to the training loss of the classifier.

Data. Here we provide additional information on the three data settings: Balanced, Imbalanced, and Imbalanced weighted. First, we consider the balanced case, where every class is represented equally in the training and test data. If the dataset is not naturally balanced, overrepresented class examples are randomly removed until balance is reached. Secondly, we consider a setting in which instances of every even-numbered class (zero-indexed) are ten times undersampled compared to their odd-numbered class counterparts, in both the train and test datasets. For computing test accuracy, we weigh the test accuracy values of every data point by the inverse relative class frequency, such that the initial importance of every class is equal, even though the rare classes contain many fewer data points. This mimics objectives in typical imbalanced data applications, where rare class instances are often considered more important than common ones [30]. For example, when training a classifier to classify defects as part

Table 2. Dataset information for the ResNet18 experiments.

dataset	# pool (balanced)	# pool (imbalanced)	# features	# classes
MNIST	53210	28815	28×28	10
FashionMNIST	59000	32000	28×28	10
SVHN	45590	24620	32×32	10
CIFAR-10	49000	26500	32×32	10

of a quality-control pipeline for industrial processes, rare defects often lead to more catastrophic failures and so are much more important to detect and classify correctly. Third, we consider the same setting, but now we additionally scale the train loss values of every data point the same way, such that our classifier model is aware of these relative weightings. As a practical and frequently used way of addressing class imbalance, this is an especially relevant nonstandard objective. Dataset details for these setting are presented in Table 2. Each benchmark dataset provides a train and test partition, to which we apply the balancing or imbalancing described above. The image classification datasets come with a predetermined train-test split, which we use in all our experiments. Due to the imbalancing step, the test data changes between the balanced and imbalanced settings (i.e., there are fewer examples of the rare classes, mimicking the train split). Due to the upweighting of rare classes during evaluation, the expected test error is similar in both settings. However, direct comparisons of the test error between settings should be treated carefully, since the datasets are not exactly equal.

AL strategies. Here we provide implementation details for our active learning strategies. Some of the implementations were adapted directly from [8], which subsamples the pool dataset to 10000 datapoints before running the acquisition strategy, to save on computation. For those methods adapted from [8] we use this subsampling implementation, as it did not affect results significantly compared to full sampling in preliminary experiments.

Uncertainty Sampling: We use standard implementations of the uncertainty sampling methods. These methods have no hyperparameters beyond the choice of strategy (Entropy, Margin, Least Confident).

HAL: For both HALUniform and HALGauss, the exploration probability is set to 0.5. Gaussian exploration in HALGauss is performed with a δ (which is related to the variance of the Normal) of 10.

CBAL: The regularisation hyperparameter λ is set to 1.0 in our experiments.

k-Center Greedy: Distances are computed using a Euclidean metric on the ResNet feature representation. This representation is the flattened activations

of the ResNet before the final Linear layer. The basic implementation was taken from [8][7].

Learning Loss: The loss prediction module takes as input the output of the four basic ResNet blocks and is trained using SGD for 200 epochs with learning rate 0.1, momentum 0.9 and weight decay 0.0005. Learning rate is decayed by a factor 10 after 160 epochs. Margin ξ and loss weight λ are both set to 1.0. Training occurs end-to-end with the ResNet, but uses a different optimiser (SGD vs. Adam). Gradients of the ResNet features feeding into the loss prediction module are detached starting at epoch 120, to increase stability. The basic implementation was taken from [8].

VAAL: VAAL is trained using batch size 128 for 100 epochs with the Adam optimiser and learning rate 0.0005. It consists of a convolutional VAE with four encoding and decoding layers, and a three-layer Multi-Layer Perceptron with hidden dimension 512 as the Discriminator. Two VAE steps with $\beta = 1.0$ are performed for each batch for every Discriminator step. The loss parameters λ_1 and λ_2 are both set to 1.0, such that the VAE and Discriminator loss are weighted equally. The basic implementation was taken from [8].

GCN: The graph network consists of two Graph Convolution layers with hidden dimension 128. Dropout with probability 0.3 is applied after the first layer. The first layer has ReLU activations, the second has Sigmoid activations that map the feature representation into the two classes: 'labelled' and 'unlabelled'. The network is trained for 200 epochs by the Adam optimiser with a learning rate of 0.001 and weight decay 0.0005. The loss hyperparameter λ is set to 1.2. For UncertainGCN, the margin s_{margin} has been set to 0.1. For CoreGCN, the k-Center Greedy implementation described above is applied to the graph feature representation. The basic implementation was taken from [8].

B.2 Myopic oracle experiments

In this section, we present some additional implementation details for the Oracle experiments of Section 4. Pseudocode for obtaining improvement scores with the Oracle is presented in Algorithm 2.

Data. The UCI binary classifications datasets used are taken from the code repository[8] corresponding to [36]. We selected the three datasets 'waveform', 'mushrooms' and 'adult', as they are still large enough for our experiments after imbalancing.

Table 3 shows dataset details for both the balanced and imbalanced settings. These datasets do not come pre-split into train and test data. We split off 200

[7] `https://github.com/razvancaramalau/Sequential-GCN-for-Active-Learning`
[8] `https://github.com/ksenia-konyushkova/LAL-RL`

Algorithm 2: Obtaining improvement scores with the ORACLE.

Data: Annotated dataset $\mathcal{D}_{annot}$, pool dataset $\mathcal{D}_{annot}$, base classifier model C with fitting method FIT(.), scoring function SCORE (evaluated on e.g. $\mathcal{D}_{test}$ or $\mathcal{D}_{val}$).
Result: List V of improvement scores.
$V \leftarrow$ empty list ;
$v \leftarrow \text{SCORE}\left(C.\text{FIT}\left(D_{annot}\right)\right)$; `/* Score classifier on simulated data */`
for $(\boldsymbol{x}, \boldsymbol{y}) \in D_{pool}$ **do**
  $v' \leftarrow \text{SCORE}\left(C.\text{FIT}\left(D_{annot} \cup (\boldsymbol{x}, \boldsymbol{y})\right)\right)$; `/* Score classifier after adding pool point */`
  append $(v' - v)$ to V ;
end
return V

Table 3. Dataset information for the myopic oracle and Neural Process experiments.

dataset	# pool (balanced)	# pool (imbalanced)	# features	# classes
waveform	2894	1411	21	2
mushrooms	7432	3907	22	2
adult	390	34	119	2
MNIST	54010	29615	728	10

points as test data and an additional 100 points as reward data for the NP experiments. These splits are controlled by a 'data seed' that we vary during our experiments to prevent bias due to dataset selection. For each of these data seeds, we further run multiple experiments with an additional random seed controlling all other randomness in our experiments. In total, we run experiments for three different data seeds, each with three different seeds, for a total of nine experiments per setting. The use of various data splits in our experiments is expected to increase global performance variance, due to larger differences in initialisation, with some data seeds being inherently more challenging than others. We see this effect on the standard deviation in the ORACLE and NP experiments. Accuracy differences for e.g. Figure 3 were computed per seed, to account for the changing train/test data distribution. We additionally run ORACLE experiments on the MNIST dataset: these experiments are only performed using the SVM classifier, as logistic regression often failed to converge within the default number of iterations.

Classifiers. Most of the AL strategies used in our previous experiments – all except KCGREEDY – are incompatible with SVM classifiers, as SVMs do not output probabilistic predictions. One could perform Platt scaling [51] to turn SVM predictions into probabilities. However, we opt to instead compare to FSCORE: a method specific to SVM classifiers that acquires datapoints closest to

the class-separating hyperplane. This method has shown strong performance in binary classification settings for both balanced and imbalanced settings [16]. It is motivated as the strategy that attempts to most rapidly reduce the version space; the space of hypotheses consistent with the observed data [37]. We use the default scikit-learn implementations [49] of logistic regression and Support Vector Machine classifiers in our experiments. Class-weighted training is performed using the 'sample weights' parameter in the built-in fit function.

AL strategies. Here we present additional implementation details for the AL strategies used in our Myopic Oracle experiments of Section 4. The implementations for Uncertainty Sampling, HAL and CBAL are identical to those described in section B.1. k-Center Greedy is implemented similarly as well, but the greedy min-max problem is solved directly on the input features, rather than on a transformed feature space.

F-Score: This strategy consists of choosing the pool point with the shortest absolute distance to the separating hyperplane of the trained SVM classifier. This approach is known to have strong performance in the binary classification setting [16]. The generalisation to the multi-class setting is not unique: for our MNIST experiments, we choose the point that has the minimum distance to any of the one-versus-rest classification boundaries.

Myopic Oracle: The myopic oracle chooses the point that maximises test accuracy after retraining, by retraining the underlying classifier on every potential added pool point. This method has no hyperparameters.

B.3 Neural process experiments

In this section, we present some additional implementation details for the Neural Process experiments of Section 5.

Neural Process model. The Neural Process implementation used in our experiments is based on the code by [14][9]. Our model consists of an Encoder and a Decoder module. The first part of the Encoder is a 1-hidden layer ReLU MLP that is weight-shared between context and target points. This MLP is applied per datapoint to either the context features $\boldsymbol{f}_{\mathcal{C}}$ or the target features $\boldsymbol{f}_{\tau}$, resulting in a datapoint-wise encoding of hidden dimension 32. The context encoding is further processed by the second part of the Encoder – a 2-layer ReLU MLP – and combined with the target encoding through an attention mechanism taken from the Image Transformer [48]. The Decoder takes the resulting representation as input together with the base target encoding and outputs a statistic representing the mean and variance of the prediction for each target datapoint. We

[9] `https://github.com/YannDubs/Neural-Process-Family`

only use the mean prediction for our active learning strategy. The model has a total of $21,924$ parameters.

The second part of the Encoder takes context label values as additional input to help predict the target label values. In our implementation, labels represent expected improvement in classification accuracy upon annotating a datapoint, and there are multiple ways to interpret this for context points. The first is to imagine the improvement corresponding to the setting where we have annotations for the entire context except the point in question (a leave-one-out type of setting). This involves computing such improvements on the reward set for the context points to construct the model input. The second is to set this label to 0 for simplicity since we do not expect much model improvement by adding the same data point twice and we already possess an annotation for the data point in question. Preliminary experimentation showed no significant differences between either training method. Our presented results correspond to the second – simplified – setting.

One may wonder why we do not use a Latent Neural Process (LNP) [19] instead of a CNP, as the former can potentially learn more complex functions. The main draw of the LNP is the ability to learn a joint $p_\theta(\boldsymbol{s}_\tau|\boldsymbol{f}_\tau;\mathcal{C})$ that does not factorise conditional on the context $\mathcal{C}$. Due to our assumption of pool point independence, this adds unnecessary complexity.

Data and training. In our experiments the context features $\boldsymbol{f}_\mathcal{C}$ and target features $\boldsymbol{f}_\tau$ are simply the normalised raw values from the dataset. Additional features – such as classifier predictions, or context class labels – can easily be incorporated as well. We observed no significant improvements using target classifier predictions in early experimentation and leave the exploration of including context labels as future work.

The Neural Process model is trained for 100 epochs by the Adam optimiser with a learning rate 0.001. We exponentially decay the learning rate by a factor 10 during those 100 epochs. The model takes the raw data features – normalised to the range $[-1, 1]$ – as input: note that it is possible to add classifier-specific features as well. The target (pool) improvement values are precomputed using the myopic oracle on the reward data $\mathcal{D}_{reward}$.

A single data 'point' during training corresponds to a full simulated active learning dataset: i.e., a set of context (annotated) points and a set of target (pool) points, all sampled uniformly from the available annotated dataset $\mathcal{D}_{\dashv\backslash\backslash\wr\sqcup}$. We group simulated AL problems of the same size together, for efficient batching with batch size 64. For sampling the AL problems, we use $Q = \{0.1, 0.2, ..., 0.8, 0.9\}$ and $N_{sim} = 300$. Preliminary experimentation showed no performance increase for larger values of N_{sim}, while using $N_{sim} = 100$ led to slight performance decreases. We did not extensively experiment with different values for Q. Our simulated AL problems all use the full number of datapoints in $\mathcal{D}_{annot}$ (i.e. no subsampling).

Training Cost Analysis: A full NP experiment (10 acquisition steps) on the UCI data takes 10-20 minutes on a 1080Ti GPU, with each acquisition step

Table 4. AL strategy AUAC and final-step test accuracy on CIFAR-10 dataset with ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels. Averages and standard deviations are computed over three seeds. The method with the highest mean performance is bolded, as well as any method whose 1 standard deviation bands include that mean.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ENTROPY	5.99 ± 0.07	0.69 ± 0.05	$\mathbf{5.11} \pm 0.04$	$\mathbf{0.61} \pm 0.02$	$\mathbf{4.84} \pm 0.13$	0.58 ± 0.01
MARGIN	6.09 ± 0.03	0.73 ± 0.00	5.02 ± 0.07	$\mathbf{0.62} \pm 0.02$	$\mathbf{4.85} \pm 0.09$	0.53 ± 0.04
LSTCONF	5.92 ± 0.16	0.74 ± 0.01	$\mathbf{5.07} \pm 0.07$	$\mathbf{0.63} \pm 0.01$	$\mathbf{4.80} \pm 0.05$	0.57 ± 0.02
KCGRDY	5.98 ± 0.02	0.72 ± 0.02	4.71 ± 0.04	0.55 ± 0.01	4.78 ± 0.02	0.56 ± 0.02
LLOSS	$\mathbf{6.19} \pm 0.06$	$\mathbf{0.75} \pm 0.01$	4.85 ± 0.04	0.61 ± 0.01	4.75 ± 0.08	$\mathbf{0.61} \pm 0.01$
VAAL	6.01 ± 0.04	0.71 ± 0.01	4.70 ± 0.05	0.55 ± 0.04	4.66 ± 0.04	0.55 ± 0.04
UNCGCN	6.02 ± 0.05	0.71 ± 0.00	4.77 ± 0.07	0.56 ± 0.01	4.69 ± 0.03	0.56 ± 0.02
COREGCN	6.05 ± 0.04	0.70 ± 0.01	4.75 ± 0.08	0.59 ± 0.01	$\mathbf{4.78} \pm 0.09$	0.59 ± 0.01
HALUNI	5.70 ± 0.03	0.63 ± 0.00	4.37 ± 0.04	0.51 ± 0.01	4.44 ± 0.06	0.55 ± 0.01
HALGAU	5.32 ± 0.04	0.60 ± 0.04	4.52 ± 0.08	0.55 ± 0.01	4.58 ± 0.13	0.54 ± 0.02
CBAL	6.02 ± 0.02	0.72 ± 0.00	4.73 ± 0.09	0.59 ± 0.01	4.63 ± 0.06	0.55 ± 0.01
RANDOM	6.00 ± 0.05	0.72 ± 0.02	4.79 ± 0.03	0.56 ± 0.02	4.72 ± 0.02	0.54 ± 0.02

taking under 2 minutes. Training the AttnCNP itself takes only 10s-15s every acquisition step. The bottleneck is in generating the data, due to the repeated retraining of the task classifier. In every acquisition step, this takes 35s-50s for the SVM classifier and 65-85s for the logistic regression classifier. Running an NP experiment for the MNIST dataset with SVM classifier takes a bit under 2 hours on the same hardware. Per acquisition step, dataset creation takes up to 10 minutes (due to the increased number of samples) and NP training about 30s.

Scalability: The primary limitation of our Neural Process approach is scalability. Supervised learning on the myopic oracle requires retraining the base classifier a large number of times during NP training, which is infeasible for large neural network models. We note that many potential acquisitions lead to (near-)zero improvement of the classifier: such data points are less interesting for LAL, but take a large fraction of computational resources during training. Strategies for spending less compute on these points may improve scalability.

C Additional results

C.1 ResNet experiments

In this section, we provide additional results for our ResNet experiments.

Table 4 shows performance on CIFAR-10 per imbalancing setting. Area Under the Acquisition Curve (AUAC) computes the area under the accuracy per

acquisition-step curve, such as those depicted in Figure 1. This is a measure of performance over the entire acquisition trajectory. We also report the final test accuracy score. As expected, performance is best across the board in the Balanced setting. Interestingly, class-weighted training (Imbalance weighted) seems mostly detrimental to performance, although the effect is less pronounced for the FashionMNIST and SVHN datasets, and reversed for MNIST (see Tables 5, 6, and 7). Note that there is no consistent best performer among the AL methods. While the three uncertainty sampling methods ENTROPY, MARGIN, and LST-CONF are strong performers across datasets, their rank-order varies with dataset, objective setting, and metric. This variation in (relative) performance across benchmarks has been previously observed in the literature [3,15,52,54,68,12].

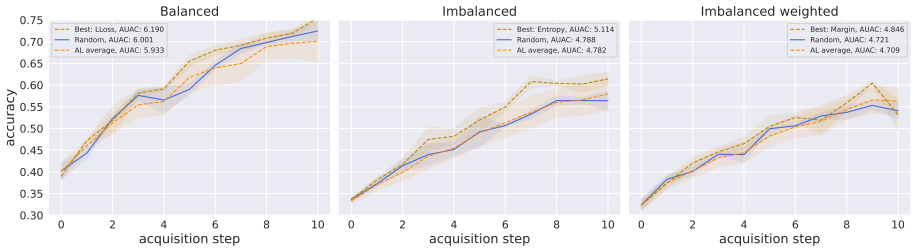

Fig. 5. Random vs. best and average of remaining AL strategies for CIFAR-10 dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels. The shaded region represents the standard deviation over three seeds.

Comparing active learning across objectives. In order to compare the performance of active learning across objectives, we once again present Figure 1 from the main text in Figure 5 above. For an ideal strategy, a single acquisition round in the Imbalanced setting should be sufficient to achieve approximately Balanced performance: since we initialise AL with 1000 points and acquire 1000 more every acquisition round, a class-balancing AL strategy could 'just' pick mostly rare classes to fully balance the dataset after one step. In practice, this is more difficult, since the labels are unknown. Still, we expect a strong active learner to achieve performance on Imbalanced not much worse than Balanced after a number of such acquisition steps, assuming sufficient rare class examples exist. For the CIFAR-10 experiments, there are 500 examples of each rare class in the training data, so exact balancing is possible until acquisition step four (5000 total datapoints). In Figure 1, we observe that the best AL strategy in the Imbalanced setting does not come close to RANDOM performance in the Balanced setting. In fact, the gap in performance widens with the number of acquisition steps, indicating that the tested active learning methods themselves perform worse in imbalanced settings than in balanced settings. This is even true of the

Table 5. AL strategy AUAC and final-step test accuracy on SVHN dataset with ResNet18 classifier, 1000 acquisitions per step and 1000 initial labels. Averages and standard deviations are computed over three seeds. The method with the highest mean performance is bolded, as well as any method whose 1 standard deviation bands include that mean.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ENTROPY	8.47 ± 0.03	$\mathbf{0.94} \pm 0.01$	$\mathbf{7.91} \pm 0.09$	0.86 ± 0.01	$\mathbf{7.72} \pm 0.14$	$\mathbf{0.89} \pm 0.01$
MARGIN	$\mathbf{8.66} \pm 0.07$	0.93 ± 0.00	$\mathbf{7.96} \pm 0.04$	0.87 ± 0.00	$\mathbf{7.76} \pm 0.11$	0.88 ± 0.01
LSTCONF	8.62 ± 0.04	0.93 ± 0.01	$\mathbf{7.93} \pm 0.08$	$\mathbf{0.87} \pm 0.01$	$\mathbf{7.82} \pm 0.05$	$\mathbf{0.88} \pm 0.01$
KCGRDY	$\mathbf{8.60} \pm 0.08$	0.92 ± 0.01	7.45 ± 0.03	0.83 ± 0.01	7.57 ± 0.12	0.85 ± 0.00
LLOSS	$\mathbf{8.64} \pm 0.03$	0.92 ± 0.00	7.32 ± 0.01	0.82 ± 0.01	7.29 ± 0.12	0.86 ± 0.01
VAAL	8.55 ± 0.07	0.91 ± 0.01	7.45 ± 0.05	0.82 ± 0.00	7.54 ± 0.11	0.85 ± 0.01
UNCGCN	8.57 ± 0.08	0.92 ± 0.00	7.40 ± 0.05	0.83 ± 0.01	7.55 ± 0.16	0.85 ± 0.01
COREGCN	8.58 ± 0.08	0.92 ± 0.01	7.31 ± 0.02	0.82 ± 0.00	7.50 ± 0.10	0.84 ± 0.01
HALUNI	8.41 ± 0.03	0.90 ± 0.00	7.06 ± 0.08	0.80 ± 0.01	7.46 ± 0.10	0.84 ± 0.00
HALGAU	8.09 ± 0.06	0.86 ± 0.01	6.58 ± 0.10	0.73 ± 0.02	6.99 ± 0.09	0.81 ± 0.01
CBAL	8.54 ± 0.05	0.91 ± 0.01	7.35 ± 0.10	0.83 ± 0.00	7.50 ± 0.15	0.84 ± 0.01
RANDOM	8.56 ± 0.06	0.91 ± 0.01	7.48 ± 0.03	0.82 ± 0.01	7.59 ± 0.08	0.85 ± 0.01

methods designed for imbalanced data (CBAL, HALUNI, HALGAU). However, note that there is a slight variation in test data between objectives due to the imbalancing step, as noted in B.1. As such, direct comparisons between these test accuracies should be treated carefully. For the other three benchmarks, active learning has slightly improved relative performance, as can be seen in Tables 5, 6, 7, and Figures 6, 7, 8. Still, the average of AL methods does not significantly outperform RANDOM in any setting and even the best strategies in imbalanced settings reach Balanced RANDOM performance in fewer than half the experiments only. In conclusion, it seems that the tested AL methods generally perform worse in imbalanced data settings than in balanced data settings, suggesting that current AL methods may be under-optimised for the former.

C.2 Oracle experiments

We refer to section C.3 for additional ORACLE results.

C.3 Neural Process experiments

In this section we provide additional Neural Process results.

Additional results: Logistic regression. In this section we provide additional results for our logistic regression experiments. First, Figure 9 shows average learning curves for the AttnCNP model for the tenth acquisition round of

Table 6. AL strategy AUAC and final-step test accuracy on FashionMNIST dataset with ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels. Averages and standard deviations are computed over three seeds. The method with the highest mean performance is bolded, as well as any method whose 1 standard deviation bands include that mean.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ENTROPY	8.94 ± 0.01	$\mathbf{0.92} \pm 0.00$	$\mathbf{8.69} \pm 0.03$	$\mathbf{0.89} \pm 0.01$	$\mathbf{8.67} \pm 0.04$	$\mathbf{0.89} \pm 0.00$
MARGIN	8.95 ± 0.01	$\mathbf{0.92} \pm 0.00$	$\mathbf{8.69} \pm 0.04$	$\mathbf{0.89} \pm 0.01$	$\mathbf{8.65} \pm 0.03$	$\mathbf{0.89} \pm 0.01$
LSTCONF	$\mathbf{8.96} \pm 0.02$	0.92 ± 0.00	$\mathbf{8.67} \pm 0.02$	$\mathbf{0.89} \pm 0.01$	$\mathbf{8.64} \pm 0.05$	$\mathbf{0.89} \pm 0.01$
KCGRDY	8.82 ± 0.01	0.90 ± 0.00	8.28 ± 0.01	0.86 ± 0.01	8.36 ± 0.05	0.86 ± 0.01
LLOSS	8.79 ± 0.02	0.90 ± 0.00	8.28 ± 0.05	0.86 ± 0.01	8.27 ± 0.04	0.86 ± 0.01
VAAL	8.83 ± 0.01	0.90 ± 0.00	8.24 ± 0.04	0.86 ± 0.01	8.29 ± 0.01	0.86 ± 0.01
UNCGCN	8.80 ± 0.02	0.91 ± 0.00	8.26 ± 0.07	0.84 ± 0.01	8.30 ± 0.06	0.86 ± 0.01
COREGCN	8.83 ± 0.02	0.90 ± 0.00	8.29 ± 0.04	0.87 ± 0.00	8.38 ± 0.02	0.86 ± 0.01
HALUNI	8.72 ± 0.02	0.89 ± 0.00	8.07 ± 0.03	0.85 ± 0.00	8.25 ± 0.05	0.86 ± 0.01
HALGAU	8.37 ± 0.04	0.86 ± 0.01	7.72 ± 0.06	0.78 ± 0.01	7.90 ± 0.06	0.81 ± 0.01
CBAL	8.83 ± 0.01	0.90 ± 0.00	8.30 ± 0.02	0.85 ± 0.01	8.31 ± 0.05	0.86 ± 0.01
RANDOM	8.82 ± 0.01	0.90 ± 0.00	8.28 ± 0.04	0.86 ± 0.00	8.32 ± 0.05	0.86 ± 0.00

the waveform dataset on all three settings. The negative log likelihood (NLL) of the AttnCNP smoothly decreases, suggesting stable learning of the model. Qualitatively similar results are observed for the other acquisition rounds.

Table 8 shows precision and recall on the rare class for the waveform dataset. Interestingly, we note that precision is favoured by the classifier in the Imbalanced setting, while recall is favoured in the Imbalanced weighted setting.

Table 9 and 10 show all remaining accuracy and AUAC results on respectively the mushrooms and adult dataset. Figures 10 and 11 show the performance as a function of acquisition step.

Additional results: SVM. Here we provide additional results for an SVM classifier. Table 11 and Figure 12 show performance on the waveform dataset with the SVM classifier. Table 13, Figure 15 and Table 14, Figure 16 show similar results for respectively the mushrooms and adult dataset. As noted previously, FSCORE consistently ranks highest of the AL methods. Since AL AVERAGE consists of only FSCORE and KCGRDY here, its strong performance is primarily driven by FSCORE. Even so, this setting seems more difficult for NP to learn, suggesting that the choice of underlying classifier is important. Figure 13 corroborates this hypothesis, although it again suggests that NP is specifically more promising for imbalanced objectives.

Figure 14 shows average learning curves for the AttnCNP model for the tenth acquisition round of the waveform dataset with SVM classifier on all three

Table 7. AL strategy AUAC and final-step test accuracy on MNIST dataset with ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels. Averages and standard deviations are computed over three seeds. The method with the highest mean performance is bolded, as well as any method whose 1 standard deviation bands include that mean. Test accuracy columns are not bolded, as nearly all methods overlap in performance.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ENTROPY	$\mathbf{9.92} \pm 0.00$	1.00 ± 0.00	$\mathbf{9.83} \pm 0.03$	0.99 ± 0.00	$\mathbf{9.86} \pm 0.02$	0.99 ± 0.00
MARGIN	$\mathbf{9.92} \pm 0.00$	1.00 ± 0.00	$\mathbf{9.84} \pm 0.03$	0.99 ± 0.00	$\mathbf{9.86} \pm 0.02$	0.99 ± 0.00
LSTCONF	$\mathbf{9.92} \pm 0.00$	1.00 ± 0.00	$\mathbf{9.83} \pm 0.03$	0.99 ± 0.00	$\mathbf{9.86} \pm 0.02$	0.99 ± 0.00
KCGRDY	9.88 ± 0.00	0.99 ± 0.00	9.74 ± 0.04	0.98 ± 0.00	9.78 ± 0.02	0.99 ± 0.00
LLOSS	9.87 ± 0.00	0.99 ± 0.00	9.71 ± 0.03	0.98 ± 0.01	9.74 ± 0.02	0.99 ± 0.00
VAAL	9.88 ± 0.00	0.99 ± 0.00	9.74 ± 0.03	0.99 ± 0.00	9.80 ± 0.03	0.99 ± 0.00
UNCGCN	9.88 ± 0.00	0.99 ± 0.00	9.75 ± 0.05	0.99 ± 0.00	9.79 ± 0.03	0.99 ± 0.00
COREGCN	9.88 ± 0.00	0.99 ± 0.00	9.75 ± 0.03	0.98 ± 0.00	9.79 ± 0.03	0.99 ± 0.00
HALUNI	9.85 ± 0.01	0.99 ± 0.00	9.68 ± 0.05	0.98 ± 0.00	9.76 ± 0.01	0.99 ± 0.00
HALGAU	9.74 ± 0.01	0.98 ± 0.00	9.51 ± 0.04	0.97 ± 0.00	9.67 ± 0.03	0.98 ± 0.00
CBAL	9.88 ± 0.01	0.99 ± 0.00	9.75 ± 0.02	0.99 ± 0.00	9.80 ± 0.02	0.99 ± 0.00
RANDOM	9.88 ± 0.00	0.99 ± 0.00	9.74 ± 0.04	0.99 ± 0.00	9.80 ± 0.02	0.99 ± 0.00

settings. The negative log likelihood (NLL) of the AttnCNP smoothly decreases, suggesting stable learning of the model. Qualitatively similar results are observed for the other acquisition rounds.

Table 12 shows precision and recall on the rare class for the waveform dataset with SVM classifier. Interestingly, we note that precision is favoured by the classifier in both imbalanced settings, except for the Oracle strategy, which favours recall for Imbalanced weighted.

We provide multiclass MNIST SVM experiments, shown in Table 15 and Figure 17. The naive FSCORE strategy – finding the datapoint closest to any of the ten one-vs-rest decision hyperplanes – experiences a strong drop in performance, indicating that less naive generalisations such as those in [37] are necessary.

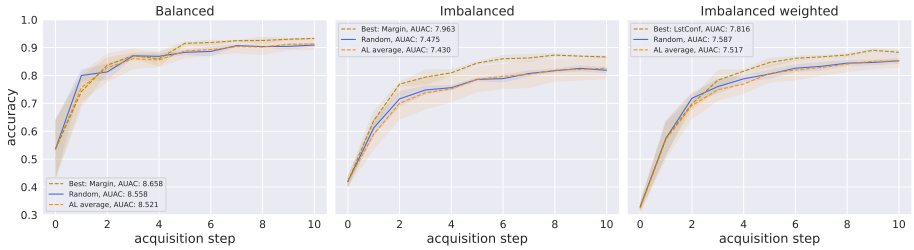

Fig. 6. Random vs. best and average of remaining AL strategies for SVHN dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels.

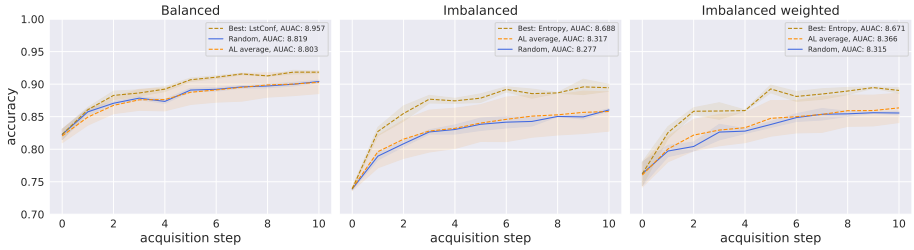

Fig. 7. Random vs. best and average of remaining AL strategies for FashionMNIST dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels.

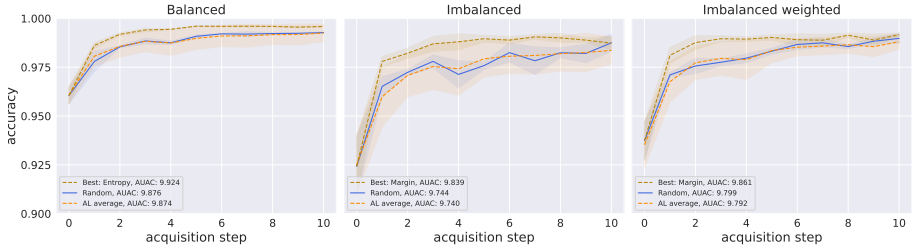

Fig. 8. Random vs. best and average of remaining AL strategies for MNIST dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels.

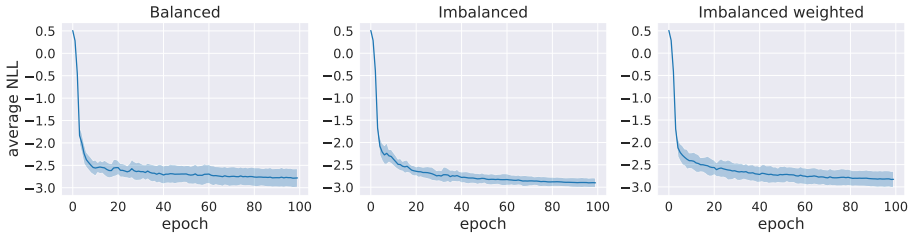

Fig. 9. Learning curves – negative log likelihood (NLL) –for the AttnCNP model for the tenth acquisition step on the waveform dataset with logistic regression classifier. Standard deviation computed over nine seeds.

Table 8. Final-step Precision and Recall per AL strategy for the rare class on the UCI waveform dataset with a logistic regression classifier. 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	Precision	Recall	Precision	Recall	Precision	Recall
ORACLE	0.95 ± 0.02	0.90 ± 0.03	0.98 ± 0.03	0.78 ± 0.09	0.70 ± 0.14	0.91 ± 0.06
UNCSAMP	0.88 ± 0.02	0.86 ± 0.04	0.95 ± 0.07	0.70 ± 0.09	0.70 ± 0.16	0.77 ± 0.06
KCGRDY	0.88 ± 0.03	0.86 ± 0.04	0.94 ± 0.07	0.68 ± 0.08	0.71 ± 0.18	0.77 ± 0.09
HALUNI	0.88 ± 0.04	0.85 ± 0.04	0.94 ± 0.08	0.62 ± 0.11	0.66 ± 0.20	0.75 ± 0.10
HALGAU	0.90 ± 0.02	0.84 ± 0.04	0.94 ± 0.07	0.64 ± 0.11	0.69 ± 0.18	0.73 ± 0.10
CBAL	0.88 ± 0.04	0.86 ± 0.04	0.95 ± 0.06	0.69 ± 0.08	0.66 ± 0.13	0.78 ± 0.07
RANDOM	0.89 ± 0.03	0.84 ± 0.05	0.95 ± 0.07	0.65 ± 0.10	0.66 ± 0.17	0.74 ± 0.10
NP	0.89 ± 0.03	0.85 ± 0.04	0.93 ± 0.08	0.67 ± 0.11	0.71 ± 0.18	0.78 ± 0.08

Table 9. AL strategy AUAC and final-step test accuracy on UCI mushrooms dataset with logistic regression classifier, 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ORACLE	**9.21** $\pm$ 0.14	**0.93** $\pm$ 0.01	**7.82** $\pm$ 0.75	**0.83** $\pm$ 0.09	**9.16** $\pm$ 0.25	**0.93** $\pm$ 0.02
UNCSAMP	8.87 $\pm$ 0.15	0.90 $\pm$ 0.01	6.35 $\pm$ 0.62	0.66 $\pm$ 0.06	8.18 $\pm$ 0.66	0.83 $\pm$ 0.07
KCGRDY	8.82 $\pm$ 0.14	0.89 $\pm$ 0.02	6.20 $\pm$ 0.39	0.63 $\pm$ 0.06	8.46 $\pm$ 0.54	0.86 $\pm$ 0.04
HALUNI	8.81 $\pm$ 0.20	0.88 $\pm$ 0.02	5.91 $\pm$ 0.54	0.59 $\pm$ 0.05	8.22 $\pm$ 0.63	0.83 $\pm$ 0.06
HALGAU	8.78 $\pm$ 0.18	0.88 $\pm$ 0.02	5.90 $\pm$ 0.54	0.59 $\pm$ 0.05	8.24 $\pm$ 0.62	0.83 $\pm$ 0.06
CBAL	8.85 $\pm$ 0.15	0.89 $\pm$ 0.01	6.07 $\pm$ 0.71	0.62 $\pm$ 0.07	8.15 $\pm$ 0.64	0.81 $\pm$ 0.07
RANDOM	8.84 $\pm$ 0.17	0.89 $\pm$ 0.01	5.99 $\pm$ 0.52	0.59 $\pm$ 0.04	8.26 $\pm$ 0.61	0.83 $\pm$ 0.07
NP	8.82 $\pm$ 0.22	0.88 $\pm$ 0.03	6.29 $\pm$ 0.55	0.67 $\pm$ 0.06	8.26 $\pm$ 0.56	0.83 $\pm$ 0.05

Table 10. AL strategy AUAC and final-step test accuracy on UCI adult dataset with logistic regression classifier, 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ORACLE	**8.02** $\pm$ 0.37	**0.82** $\pm$ 0.04	**6.69** $\pm$ 0.59	**0.68** $\pm$ 0.05	**7.16** $\pm$ 0.63	**0.72** $\pm$ 0.06
UNCSAMP	7.46 $\pm$ 0.43	0.75 $\pm$ 0.04	6.38 $\pm$ 0.41	0.66 $\pm$ 0.05	6.85 $\pm$ 0.50	0.69 $\pm$ 0.06
KCGRDY	7.47 $\pm$ 0.40	0.74 $\pm$ 0.04	6.38 $\pm$ 0.52	0.65 $\pm$ 0.06	6.77 $\pm$ 0.66	0.68 $\pm$ 0.07
HALUNI	7.48 $\pm$ 0.49	0.75 $\pm$ 0.04	6.30 $\pm$ 0.47	0.63 $\pm$ 0.04	6.66 $\pm$ 0.55	0.66 $\pm$ 0.06
HALGAU	7.44 $\pm$ 0.45	0.74 $\pm$ 0.05	6.26 $\pm$ 0.47	0.62 $\pm$ 0.05	6.64 $\pm$ 0.55	0.66 $\pm$ 0.05
CBAL	7.47 $\pm$ 0.42	0.74 $\pm$ 0.04	6.39 $\pm$ 0.42	0.66 $\pm$ 0.05	6.79 $\pm$ 0.55	0.69 $\pm$ 0.07
RANDOM	7.53 $\pm$ 0.41	0.75 $\pm$ 0.05	6.33 $\pm$ 0.52	0.63 $\pm$ 0.05	6.61 $\pm$ 0.59	0.67 $\pm$ 0.06
NP	7.46 $\pm$ 0.47	0.75 $\pm$ 0.05	6.35 $\pm$ 0.50	0.64 $\pm$ 0.06	6.80 $\pm$ 0.58	0.69 $\pm$ 0.07

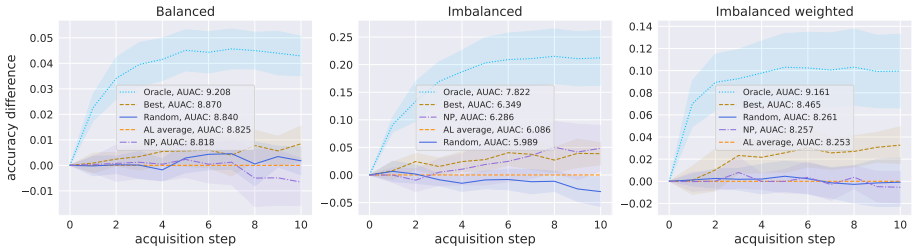

Fig. 10. Relative performance of acquisition strategies for mushrooms dataset and logistic regression classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). Shaded region represents twice the standard error of the mean over nine seeds.

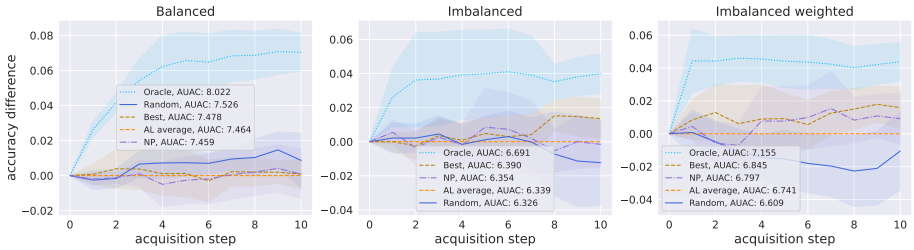

Fig. 11. Relative performance of acquisition strategies for adult dataset and logistic regression classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). Shaded region represents twice the standard error of the mean over nine seeds.

Table 11. AL strategy AUAC and final-step test accuracy on UCI waveform dataset with SVM classifier, 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ORACLE	$\mathbf{9.22} \pm 0.10$	$\mathbf{0.93} \pm 0.01$	$\mathbf{8.26} \pm 0.46$	0.84 ± 0.05	$\mathbf{9.30} \pm 0.26$	$\mathbf{0.95} \pm 0.02$
FSCORE	8.87 ± 0.15	0.89 ± 0.02	8.10 ± 0.53	$\mathbf{0.85} \pm 0.05$	8.71 ± 0.47	0.88 ± 0.06
KCGRDY	8.85 ± 0.14	0.88 ± 0.01	7.93 ± 0.56	0.81 ± 0.06	8.57 ± 0.45	0.87 ± 0.03
RANDOM	8.86 ± 0.16	0.89 ± 0.02	7.47 ± 0.53	0.75 ± 0.06	8.54 ± 0.51	0.86 ± 0.05
NP	8.85 ± 0.16	0.89 ± 0.02	7.75 ± 0.60	0.80 ± 0.07	8.50 ± 0.61	0.86 ± 0.06

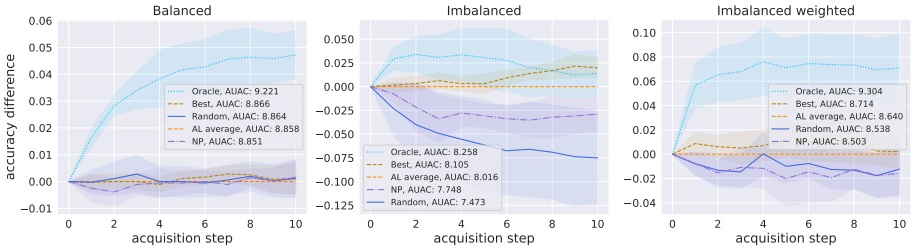

Fig. 12. Relative performance of acquisition strategies for waveform dataset and SVM classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). Shaded region represents twice the standard error of the mean over nine seeds.

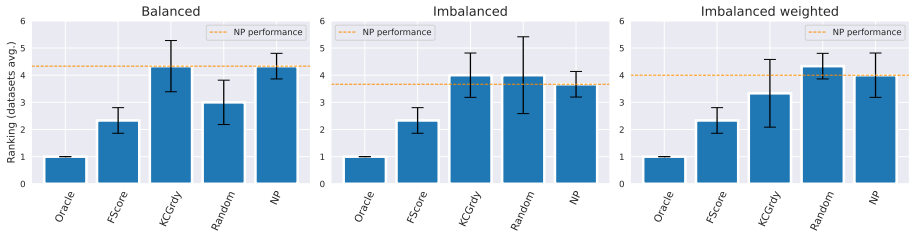

Fig. 13. Relative AUAC rank of AL strategies averaged over the three UCI datasets for SVM. Standard deviation of this rank is denoted by the error bars.

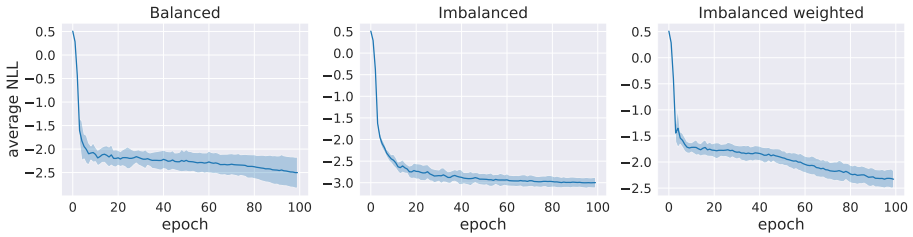

Fig. 14. Learning curves – negative log likelihood (NLL) –for the AttnCNP model for the tenth acquisition step on the waveform dataset with SVM classifier. Standard deviation computed over nine seeds.

Table 12. Final-step Precision and Recall per AL strategy for the rare class on the UCI waveform dataset with an SVM classifier. 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	Precision	Recall	Precision	Recall	Precision	Recall
ORACLE	0.98 ± 0.01	0.89 ± 0.03	0.99 ± 0.02	0.68 ± 0.10	0.86 ± 0.11	0.91 ± 0.05
FSCORE	0.92 ± 0.04	0.86 ± 0.04	0.98 ± 0.03	0.69 ± 0.10	0.80 ± 0.06	0.77 ± 0.12
KCGRDY	0.94 ± 0.02	0.82 ± 0.04	0.99 ± 0.03	0.61 ± 0.11	0.81 ± 0.20	0.77 ± 0.09
RANDOM	0.95 ± 0.03	0.82 ± 0.04	0.99 ± 0.03	0.50 ± 0.11	0.89 ± 0.12	0.73 ± 0.11
NP	0.94 ± 0.03	0.83 ± 0.04	0.99 ± 0.03	0.59 ± 0.13	0.81 ± 0.18	0.74 ± 0.13

Table 13. AL strategy AUAC and final-step test accuracy on UCI mushrooms dataset with SVM classifier, 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ORACLE	$\mathbf{9.33} \pm 0.16$	$\mathbf{0.94} \pm 0.02$	$\mathbf{6.03} \pm 0.87$	$\mathbf{0.63} \pm 0.10$	$\mathbf{9.00} \pm 0.18$	$\mathbf{0.91} \pm 0.01$
FSCORE	9.19 ± 0.13	0.93 ± 0.02	5.81 ± 0.53	0.62 ± 0.08	8.55 ± 0.44	0.86 ± 0.04
KCGRDY	9.14 ± 0.19	0.92 ± 0.02	5.14 ± 0.14	0.53 ± 0.03	8.68 ± 0.29	0.88 ± 0.02
RANDOM	9.13 ± 0.15	0.92 ± 0.01	5.06 ± 0.08	0.51 ± 0.01	8.50 ± 0.48	0.85 ± 0.04
NP	9.06 ± 0.20	0.90 ± 0.02	5.63 ± 0.51	0.60 ± 0.06	8.52 ± 0.43	0.85 ± 0.04

Table 14. AL strategy AUAC and final-step test accuracy on UCI adult dataset with SVM classifier, 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	Balanced		Imbalanced		Imbalanced weighted	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ORACLE	**8.00** $\pm$ 0.38	**0.81** $\pm$ 0.04	**5.30** $\pm$ 0.37	**0.53** $\pm$ 0.04	**7.67** $\pm$ 0.54	**0.78** $\pm$ 0.05
FSCORE	7.64 $\pm$ 0.41	0.77 $\pm$ 0.04	5.25 $\pm$ 0.36	0.52 $\pm$ 0.04	7.28 $\pm$ 0.51	0.74 $\pm$ 0.05
KCGRDY	7.62 $\pm$ 0.41	0.76 $\pm$ 0.04	5.19 $\pm$ 0.32	0.52 $\pm$ 0.04	7.20 $\pm$ 0.67	0.72 $\pm$ 0.07
RANDOM	7.66 $\pm$ 0.43	0.77 $\pm$ 0.04	5.25 $\pm$ 0.35	0.53 $\pm$ 0.04	7.20 $\pm$ 0.60	0.72 $\pm$ 0.06
NP	7.62 $\pm$ 0.41	0.76 $\pm$ 0.04	5.25 $\pm$ 0.36	0.53 $\pm$ 0.04	7.26 $\pm$ 0.61	0.73 $\pm$ 0.06

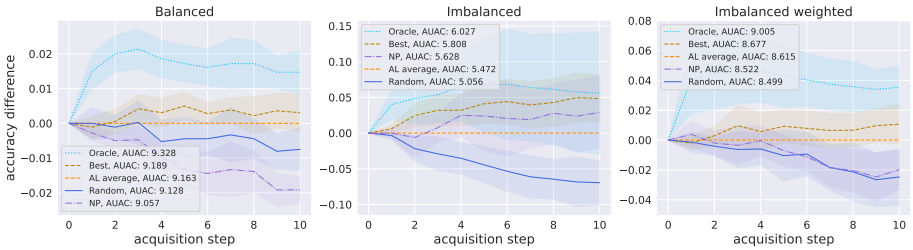

Fig. 15. Relative performance of acquisition strategies for mushrooms dataset and SVM classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). Shaded region represents twice the standard error of the mean over nine seeds.

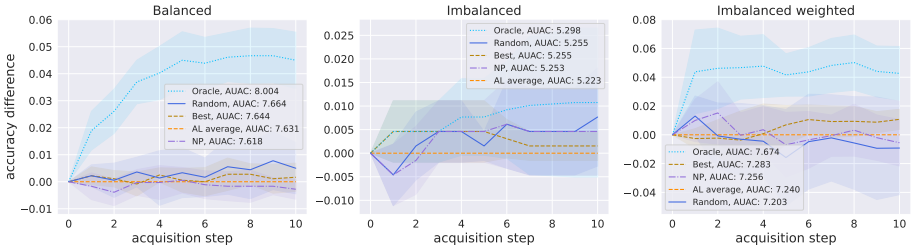

Fig. 16. Relative performance of acquisition strategies for adult dataset and SVM classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). Shaded region represents twice the standard error of the mean over nine seeds.

Table 15. AL strategy AUAC and final-step test accuracy on MNIST dataset with SVM classifier, 1 acquisition per step, and 100 initial labels. Averages and standard deviations are computed over nine seeds.

Strategy	**Balanced**		**Imbalanced**		**Imbalanced weighted**	
	AUAC	Test acc.	AUAC	Test acc.	AUAC	Test acc.
ORACLE	**8.39** $\pm$ 0.15	**0.89** $\pm$ 0.01	**5.24** $\pm$ 0.50	**0.57** $\pm$ 0.07	**7.68** $\pm$ 0.40	**0.83** $\pm$ 0.05
FSCORE	7.48 $\pm$ 0.31	0.74 $\pm$ 0.03	4.27 $\pm$ 0.19	0.43 $\pm$ 0.03	4.10 $\pm$ 0.94	0.39 $\pm$ 0.10
KCGRDY	7.50 $\pm$ 0.24	0.75 $\pm$ 0.03	4.29 $\pm$ 0.19	0.43 $\pm$ 0.02	4.60 $\pm$ 0.48	0.49 $\pm$ 0.04
RANDOM	7.53 $\pm$ 0.27	0.75 $\pm$ 0.03	4.24 $\pm$ 0.18	0.43 $\pm$ 0.02	3.60 $\pm$ 0.53	0.34 $\pm$ 0.06
NP	7.52 $\pm$ 0.29	0.75 $\pm$ 0.03	4.28 $\pm$ 0.13	0.43 $\pm$ 0.01	4.85 $\pm$ 0.61	0.47 $\pm$ 0.05

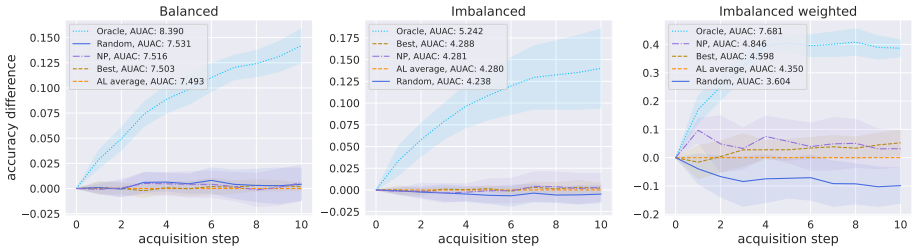

Fig. 17. Relative performance of acquisition strategies for MNIST dataset and SVM classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). Shaded region represents twice the standard error of the mean over nine seeds.