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].[6]

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[7], 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

[6] https://github.com/razvancaramalau/Sequential-GCN-for-Active-Learning
[7] 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].[8] 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

[8] 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].

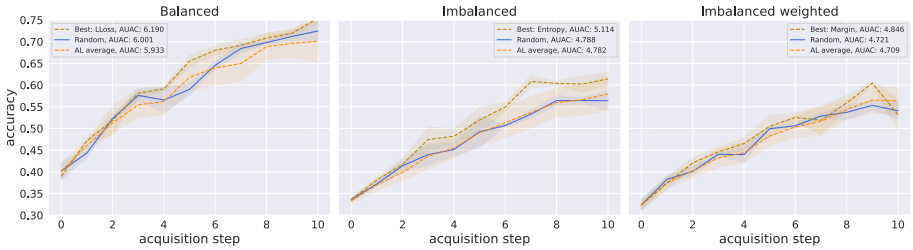

Fig. 5. Random vs. best and average of remaining AL strategies for CIFAR-10 dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels. The shaded region represents the standard deviation over three seeds.

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	**9.92** ± 0.00	1.00 ± 0.00	**9.83** ± 0.03	0.99 ± 0.00	**9.86** ± 0.02	0.99 ± 0.00
MARGIN	**9.92** ± 0.00	1.00 ± 0.00	**9.84** ± 0.03	0.99 ± 0.00	**9.86** ± 0.02	0.99 ± 0.00
LSTCONF	**9.92** ± 0.00	1.00 ± 0.00	**9.83** ± 0.03	0.99 ± 0.00	**9.86** ± 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.

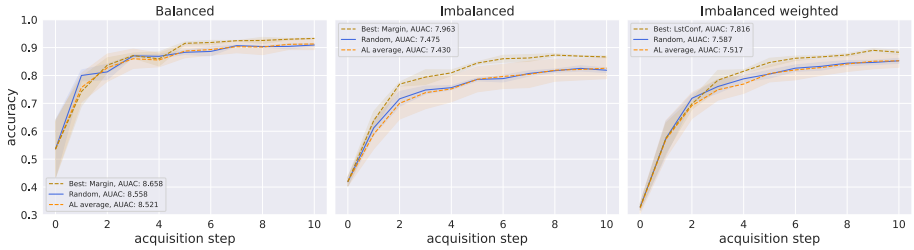

Fig. 6. Random vs. best and average of remaining AL strategies for SVHN dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels.

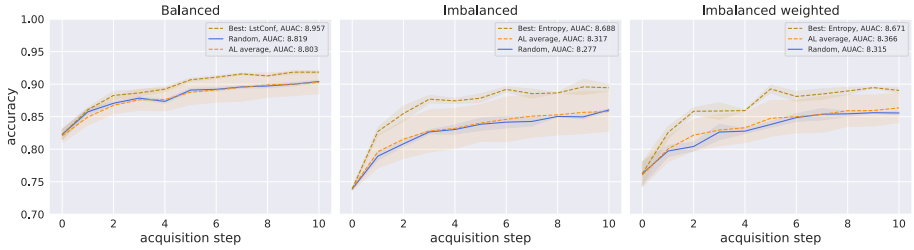

Fig. 7. Random vs. best and average of remaining AL strategies for FashionMNIST dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels.

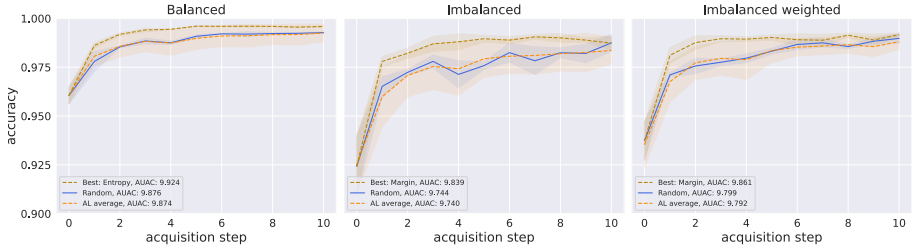

Fig. 8. Random vs. best and average of remaining AL strategies for MNIST dataset and ResNet18 classifier, 1000 acquisitions per step, and 1000 initial labels.

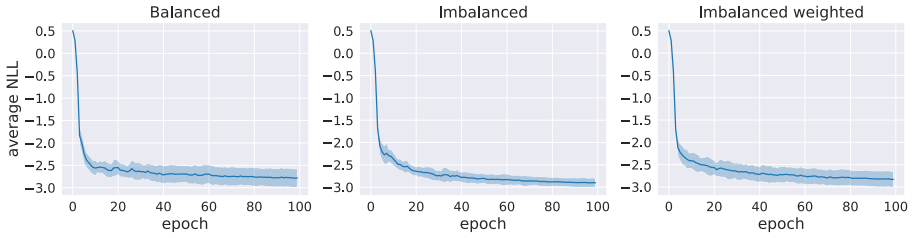

Fig. 9. Learning curves – negative log likelihood (NLL) –for the AttnCNP model for the tenth acquisition step on the waveform dataset with logistic regression classifier. Standard deviation computed over nine seeds.

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

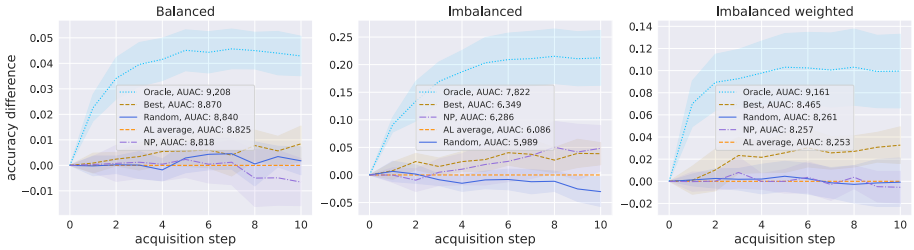

Fig. 10. Relative performance of acquisition strategies for mushrooms dataset and logistic regression classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies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

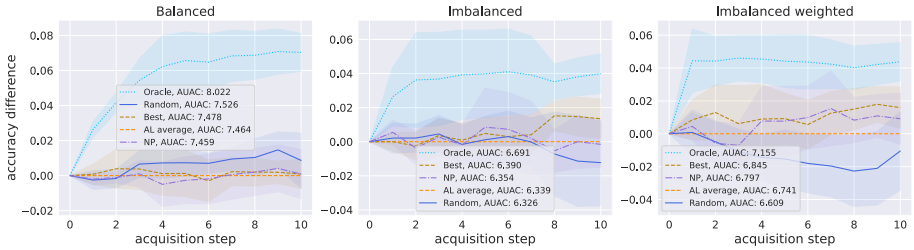

Fig. 11. Relative performance of acquisition strategies for adult dataset and logistic regression classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (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	**9.22** $\pm$ 0.10	**0.93** $\pm$ 0.01	**8.26** $\pm$ 0.46	0.84 $\pm$ 0.05	**9.30** $\pm$ 0.26	**0.95** $\pm$ 0.02
FSCORE	8.87 $\pm$ 0.15	0.89 $\pm$ 0.02	8.10 $\pm$ 0.53	**0.85** $\pm$ 0.05	8.71 $\pm$ 0.47	0.88 $\pm$ 0.06
KCGRDY	8.85 $\pm$ 0.14	0.88 $\pm$ 0.01	7.93 $\pm$ 0.56	0.81 $\pm$ 0.06	8.57 $\pm$ 0.45	0.87 $\pm$ 0.03
RANDOM	8.86 $\pm$ 0.16	0.89 $\pm$ 0.02	7.47 $\pm$ 0.53	0.75 $\pm$ 0.06	8.54 $\pm$ 0.51	0.86 $\pm$ 0.05
NP	8.85 $\pm$ 0.16	0.89 $\pm$ 0.02	7.75 $\pm$ 0.60	0.80 $\pm$ 0.07	8.50 $\pm$ 0.61	0.86 $\pm$ 0.06

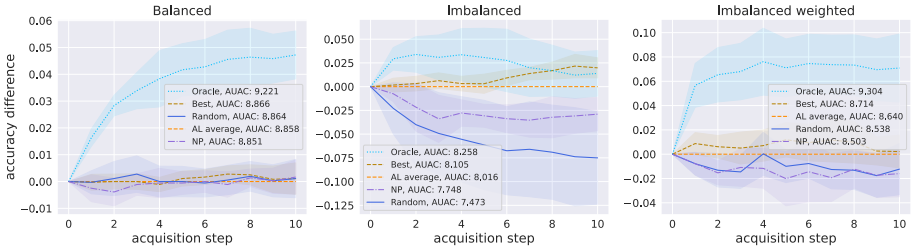

Fig. 12. Relative performance of acquisition strategies for waveform dataset and SVM classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE).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

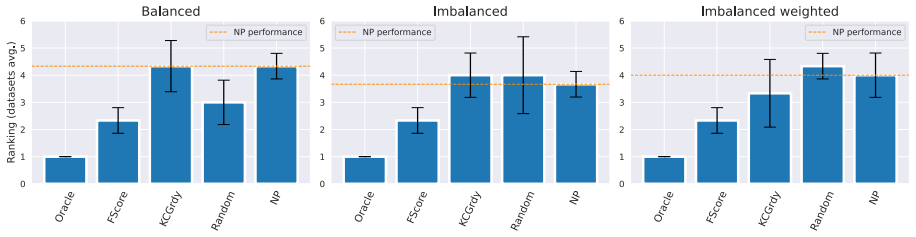

Fig. 13. Relative AUAC rank of AL strategies averaged over the three UCI datasets for SVM. Standard deviation of this rank is denoted by the error bars.

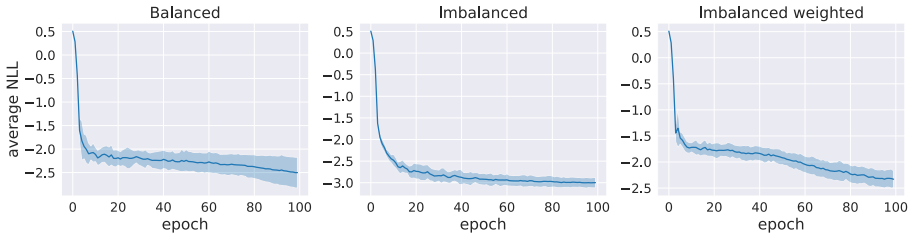

Fig. 14. Learning curves – negative log likelihood (NLL) –for the AttnCNP model for the tenth acquisition step on the waveform dataset with SVM classifier. Standard deviation computed over nine seeds.

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

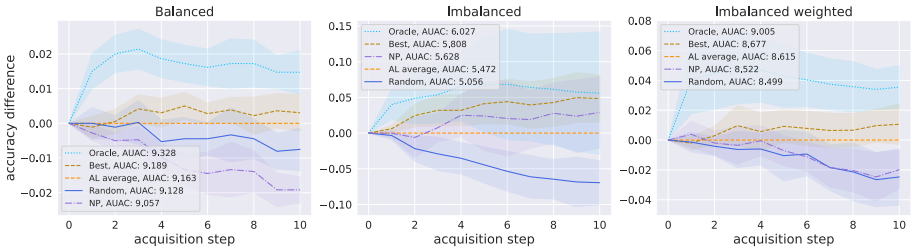

Fig. 15. Relative performance of acquisition strategies for mushrooms dataset and SVM classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE).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

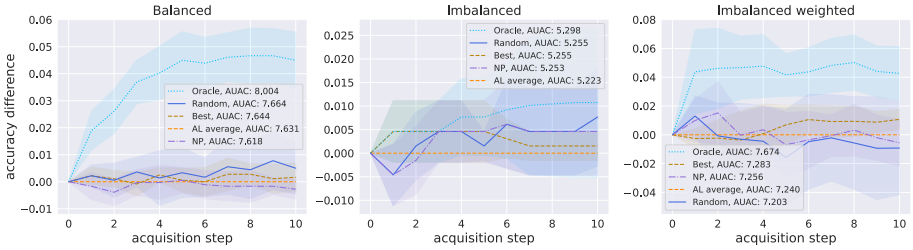

Fig. 16. Relative performance of acquisition strategies for adult dataset and SVM classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). Shaded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

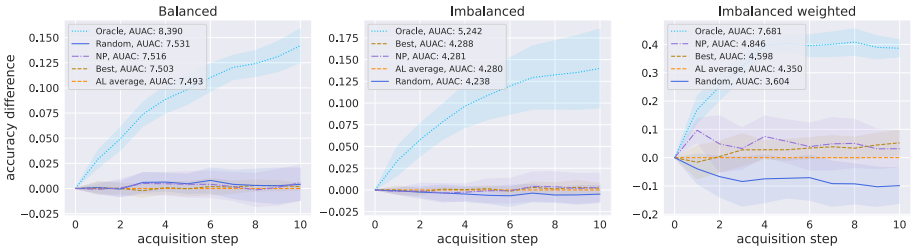

Fig. 17. Relative performance of acquisition strategies for MNIST dataset and SVM classifier, 1 acquisition per step, and 100 initial labels. Accuracy differences of RANDOM, ORACLE, NP and the best remaining AL strategy (BEST) are computed w.r.t. the average of remaining AL strategies (AL AVERAGE). Shaded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