Active Learning Strategies for Long Horse Tails

H.F. (FERDINAND) DE JONG, University of Twente, The Netherlands

Fig. 1. Horse with long-tailed distribution

Recognizing symptoms of animal disease with machine learning can be improved by balancing class distributions of animal activity, which frequently are long-tailed. Uncertainty Sampling and Disagreement-based Sampling strategies of active learning, as well as Density Weighting and a novel Pragmatic Balance approach are evaluated on their resulting class distributions in this research. This is done by applying them to a dataset of horse accelerometer data. A combination of these approaches is shown to have a significant effect in achieving a balanced training set, by finding more instances of rare tail classes and reducing the amount of instances of common head classes in the training set. Additionally, general model performance increases noticeably with these methods.

Additional Key Words and Phrases: Active Learning, Animal Activity Recognition, Long-Tailed Distribution, Open Set Recognition, Sustainability of AI

1 INTRODUCTION

Horses have been around humans for thousands of years. Since early civilization, man has used domesticated horses for various purposes [20]. The main concern for horse health is the colic, a condition of the intestines that can be - if not treated - fatal [2]. An important symptom that indicates that a horse may have a colic is frequent rolling. To detect this rolling, Animal Activity Recognition (AAR) may be able to lend a helping hand. Wearable Inertial Measurement Units (IMU) can be attached non-invasively to collect motion data, which in turn can be labeled and fed to a machine learning model. This model can then classify and monitor horse activity to detect this grave affliction timely [14].

This may sound simple, but some problems arise in practise. Collection of labeled data is considerably more expensive than the collection of unlabeled data, as the process of labelling manually requires a human to sit behind a screen for a long time. A modification of LabelStudio, proposed by Kamminga et al., which attempts to speed up the labeling process of IMU data, still requires a human to look at the screen for the duration of the samples set to be labeled [13].

Thus, because human-labeled samples are expensive, one would want to use techniques that produce a well-performing model with little labeled data. One such technique is active learning. In this paradigm, based on the eponymous education technique, an algorithm determines the next sample(s) for the model to train on. This algorithm takes into account model predictions on unlabeled samples, or featural (dis)similarities between labeled and unlabeled samples. Several different active learning strategy types exist, such as uncertainty sampling (UNCS), disagreement-based sampling (DBS) and information density sampling (IDS) [23].

Another problem that arises is the fact that collected data shows a long-tailed distribution [12]. Some activity classes are very common (head classes). Other classes (tail classes) occur rarely. Rolling is such a class. If the training data follows the distribution in the total collected data, the model is going to perform poorly when classifying these tail classes [9]. Because active learning makes a motivated selection of the unlabeled set, a more balanced labeled set may be constructed, where tail classes are more common, and head classes less so.

In this research, these methods of active learning, along with a pragmatical take on active learning in the context of distribution balancing, will be compared by applying them to a IMU dataset, gathered from domesticated horses [12]. To find a suitable model to apply these techniques to, first an established algorithm and a new algorithm will be compared on resource usage and speed, as well as performance. In the second part of this research one of these models will be applied with several active learning strategies, in order to find a model that can achieve a (more) balanced class distribution while learning from a set of human-annotated samples.

2 PREVIOUS RELATED WORK

The field of active learning has been surveyed by Settles (2010) [23]. He identifies several types of strategies, such as uncertainty sampling (UNCS), disagreement-based sampling (DBS, also known as query-by-committee) and density-weighted sampling (DWS). UNCS methods pick samples which the model is least confident about

TScIT 41, July 5, 2024, Enschede, The Netherlands

© 2024 University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

when sampling. DBS methods use a committee of models, and pick the samples that the models disagree the most on. DWS methods attempt to find samples that are more interesting to the model, as sample in dense regions of the feature space are more likely to be picked. In a paper from 2008, Settles and Craven show that DWS can outperform these other strategies on accuracy [24]. Another significant finding in this study is that DBS is resource-intensive, as it trains several models at the same time. Spink et al. (2022) compare several types of UNCS and DBS strategies applied to IMU data, but find that only DBS methods provide a (small) advantage over random sampling in terms of accuracy, where the advantage is more pronounced at low amounts of labels [27]. This study did not cover other methods of active learning.

The main starting point of this research has been the dissertation of J.W. Kamminga: *Hiding in the Deep: Online Animal Activity Recognition using Motion Sensors and Machine Learning*, in which the authors contribute in several ways to the field of AAR [14]. One such contribution is the comparison of several machine learning algorithms, such as k-NN, Naive Bayes and Support Vector Machines (SVM). It identified that although the tested SVM algorithm scored the highest on performance statistics, such as the F_1-score, the Naive Bayes algorithm did not lag far behind, with 200 times faster execution and significantly less memory usage. Some recent algorithms were not tested, such as the Random Forest algorithm, which consists of many different decision trees, and the LSTM algorithm, which is a type of Recurrent Neural Network designed to capture long-term trends in data. Huveneers (2021) recommended an LSTM, along with a variant (MLSTM-FCN), to be used on AAR IMU data, as they performed well on accuracy and F_1-score, where the MLSTM-FCN was also much faster during training [11]. One study booked particularly good results applying a Random Forest model to sheep IMU data (accuracy > 98%), but did not compare this performance to other algorithms [15].

Van Wynsberghe has contributed significantly to the field of Sustainability and AI, separating the concerns of AI for Sustainability and Sustainability of AI, in her 2021 opinion paper [28]. She calls on AI developers and researchers to track environmental impact and to minimize this impact. Sustainability is an important value, and finding ways to cut on resource demands in machine learning is one way to contribute to this.

Some investigation has been done on how to deal with imbalanced class distributions in data. For example, Buda et al. (2018) use oversampling and undersampling, techniques that discard some data in order to generate a dataset with a (more) balanced class distribution [4]. The idea here is that a more balanced class distribution in the training set would lead to a model that better predicts classes in the tail of the class distribution. Kamminga et al. object to this method in their future research section, as data would be disregarded that is otherwise perfectly useful.

This research also taps into the topic of Open Set Recognition (OSR) or Open World Recognition, areas of machine learning that take into consideration the presence of unseen and unknown classes in the data. These fields have been surveyed by Geng et al. (2020) and Mahdavi et al. (2021) [8, 18]. Geng et al. identify active learning in combination with OSR as a future research option. Some research in this area already exists, such as this study by Luo et al., where two active learning (UNCS) strategies are compared on classification of plankton, in an OSR setting [17]. Similarly, Liu and Huang propose a strategy making use of active learning to explore an open set [16]. Both studies focus on performance statistics, and no mention is made of the resulting balancing in the training set. Also related to this research are the topics of Anomaly Detection and Out-Of-Distribution (OOD) detection, as treated in Hogeweg et al. (2024) [10]. These approached attempt to detect samples that are not of known classes and classify them as such.

3 PROBLEM STATEMENT

Although active learning strategies have often been compared to the traditional random sampling methods (for a performance comparison on similar data, see: Spink et al. (2022) [27]), such comparisons have focused on performance metrics like the F_1-score and accuracy, rather than the resulting class distribution in the labeled data and its balance. This gap in existing literature gives rise to the research question below. To answer this research question, a base model is needed that trains and inferences quickly, in order to test the active learning strategies, which is what the first sub-question is for. The second sub-question then returns to the main question and builds upon the answers to the first sub-question.

3.1 Research question

How do human-in-the-loop learning methodologies affect the class distributions of human-annotated IMU data with long-tailed source data?

3.2 Sub-question 1

How do machine learning algorithms compare on speed and resource usage during training and inference, when applied to IMU data?

3.3 Sub-question 2

How do strategies of active learning and auxiliary approaches affect the class distribution of human-annotated IMU data in a long-tailed dataset?

4 METHODOLOGY

The setup of both experiments will be detailed in this section. Some theory behind the experiments will also be laid out. Experiment 1 attempts to answer research sub-question 1, and contributes to some set-up decisions for Experiment 2, which attempts to answer research sub-question 2.

4.1 Experiment 1

In order to field a fast and well-performing model for Experiment 2, two algorithms have been compared by measuring execution times and taking performance statistics. The Naive Bayes algorithm was chosen as Kamminga et al. identified it as a very fast and reasonably accurate algorithm [14]. Furthermore, a Random Forest algorithm was tested as it was not accounted for in the comparison of Kamminga et al., while it was proven to be accurate in Kleantheous et al. [14, 15].

Maximum
75th percentile
Median
25th percentile
Minimum
Mean
Kurtosis
Skewness
Mean low pass filtered signal
Mean rectified high pass signal

Table 1. Extracted features

4.1.1 Data processing. The data used consists of about 2.5 hours of recorded IMU data. This data was processed into frames of 2 seconds, resulting in 5264 frames. These frames describe 7 different activities: walking, trotting, scratch-biting, head-shake, grazing, running, standing. The activity class distribution is very much long-tailed; scratch-biting occurs just on one frame, while the horse is walking on 2142 frames. Features were extracted from the threedimensional acceleration component (the length of the x, y and z acceleration vectors combined) according to the statistical summary provided by Kamminga et al. [14]. Some features have been removed, as they would provide no information, such as the zero-crossing rate. (The threedimensional acceleration component is always positive, so no zero-crossings occur.) The features used are shown in Table 1. A more detailed explanation is available in Kamminga et al.

1250 samples (25%) were used as validation data, while 3750 samples were used as training data [14]. Both algorithms that were tested were taken from the Python library scikit-learn [21]. The Naive Bayes algorithm was supplied by the `naive_bayes.GaussianNB` module. The Random Forest algorithm was taken from the `ensemble.RandomForestClassifier` module.

4.1.2 Measuring methods. Both the training time and the inference time were measured. Since a fast model that predicts inaccurately is useless, accuracy and F_1-score were also measured on the 1250 test samples. The F_1-score was taken 'weighted', which indicates that the F_1-score was calculated for every label and then averaged with more frequent labels being weighted proportionately higher. The 'micro' F_1-score would be the same as the accuracy in this context, and 'macro' F_1-score would weigh the performance on rare classes disproportionately [22].

Timing measurements were taken using the in-built Python function `time.perf_counter_ns()`, which uses a clock with the highest resolution available for a short period of time. Previously used alternatives like Scalene and PyRAPL provide more information about memory and CPU usage or energy usage, but are unable to capture information in the tiny amounts of time the model takes to train or inference, or work on Linux only [1, 3, 5, 6]. Still, timing measurements are meaningful, as execution time generally correlates with with resource and energy usage [19].

Strategy	**Score**
Random Sampling	$U(0.9999, 1)$
Lowest Confidence Sampling (LCS)	$1 - P(y^*\|x)$
Margin of Confidence Sampling (MCS)	$1 - P(y_1^*\|x) + P(y_2^*\|x)$
Confidence Entropy Sampling (CES)	$-\sum_{i=1}^{n} P(x_i) \log P(x_i)$
Maximum Disagreement Sampling (MDS)	$\sum_i P(i) \log \frac{P(i)}{Q(i)}$

Table 2. Active learning base strategies

4.2 Experiment 2

4.2.1 Active learning base strategies. In experiment 2, several strategies of active learning are used to select samples for the model, and are evaluated on their effectiveness in balancing the class distribution. The model is trained with a base number of samples, after which it predicts the rest of the sample set. This sample set is similar to the sample set of experiment 1, but is taken from another horse, is based on a window of 1 second (200 measurements), and contains more rare classes, as seen in Figure 5. The sample(s) with the highest score given to them by the active learning algorithm is picked to be included in the training set for the model. After a batch of samples is added to the training set, the model is trained again with the training set. A small batch size is more expensive, as the model will be trained more often for the same amount of total samples added. It does mitigate some problems that arise with larger batches, such as high overlap between the samples in a batch [23]. With the retrained model, the remaining samples are evaluated, and the active learning strategy is again used to determine the most interesting sample(s).

The base strategy is random sampling, in which the score is just a random number between 0.9999 and 1. The next strategy is Lowest Confidence Sampling (LCS), in which the score increases as the probability of the most probable label decreases. The formula is shown in Table 2. Here all formulae for the base strategies are shown. $P(y^*|x)$ stands for the probability assigned to the most probable label (y^*). For Margin of Confidence Sampling, the score increases as the probabilities of the two most probable labels are closer to one another. Samples with the smallest margin are thus considered the most interesting. In Confidence Entropy Sampling (CES), the measure of entropy is used to determine whether a sample is of interest to the model [25]. A uniform distribution over the labels would be considered the most interesting, and would receive a score of 1. In Maximum Disagreement Sampling, two or more models are trained with the same training set. The relative entropy (or Kullback-Leibler divergence) between their probability distributions is used as score. In this experiment, a Random Forest model and a Naive Bayes model are used.

4.2.2 Density weighting. Additionally, methods where a score modifier is applied to the base score (from the active learning strategy) are tested. One such modifier is the density weighting score as proposed in Settles and Craven [24]. The base score is multiplied with

```
Require: sample_pool
  training_samples ⟵ initial samples from sample_pool
  fit model with training_samples
  for $i = 1$ to iterations do
      scores ⟵ predict scores for sample_pool
      move samples from sample_pool to training_samples
       based on scores
      fit model with updated training_samples
  end for
```

Fig. 2. Simplified Active Learning Loop

a density score as shown below:

$$\left(\frac{1}{U}\sum_{u=1}^{U}\text{sim}(x, x_u)\right)^{\beta}$$

A similarity score is given to all pairs of samples, and for every sample the similarity scores of its pairs is averaged. (In the formula, these pair scores are $\text{sim}(x, x_u)$, and U is the number of all samples except x.) The similarity score that was used in this experiment (as Settles and Craven advise [24]) is the cosine similarity, defined as:

$$\text{sim}_{\cos}(x, x_u) = \frac{\vec{x} \cdot \vec{x_u}}{\|\vec{x}\| \times \|\vec{x_u}\|}$$

Here the dot product of the samples' feature vectors is divided by the product of their magnitudes. This resulting density score (to a factor β) is then multiplied with the existing score. This density weighting approach was intended to find samples that were in dense contested area of the feature space, so it is expected to select more of the common classes. This is why negative β-values will also be tested. This score will be referred to as DW^{β} in the rest of the paper, with the β-value specified in the superscript.

4.2.3 Pragmatic balancing. Lastly, a pragmatic approach will be tested, where the base score determined by the active learning strategy is multiplied with a factor that depends on the existing distribution of classes in the training set. This score is formulated as follows:

$$(1 - \frac{c_{samplelabel}}{n})^{\gamma}$$

Here, c is the number of occurrences of the sample's label in the training set. The label of the sample is assumed to be the label that the model predicts, as the actual label is not known in an active learning scenario. n stands for the total number of samples in that set. A power γ is applied to tune the influence of this Pragmatic Balance score. This score will be referred to as PB^{γ} in the rest of this paper, with the γ-factor specified in the superscript.

4.2.4 Evaluation scoring. To evaluate the experiment, the relative frequencies of the classes will be measured, and to get a sense of the performance of the trained model, it is tested with the remaining data. The class distribution balance can be represented in a single number using the normalized Shannon-Wiener index [14, 26]. This measure calculates the Shannon entropy as in CES for the relative frequencies, and balances it by dividing it by the logarithm of the number of classes. The resulting metric thus ranges between 0 (one

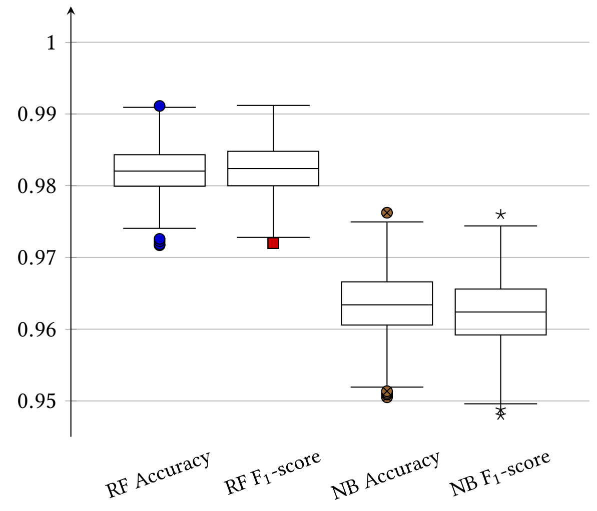

Fig. 3. Accuracy and F_1-score

class takes all) and 1 (uniformly balanced). The elaborate formula is stated below:

$$H_{norm} = \frac{H}{\log k} = \frac{-\sum_{i=1}^{k}\left(\frac{c_i}{n}\log\frac{c_i}{n}\right)}{\log k}$$

Here k is the number of classes, and c_i the amount of samples for the class, and n the total amount of samples. To get a grasp of the performance of the model, the model will be tested with the unseen samples, from which (as described in Experiment 1 (4.1.2)) the accuracy and (weighted) F_1-score.

5 EXPERIMENTS

In this section the execution and results of the experiments will be detailed, followed by a brief discussion. Both experiments were performed on the author's laptop (HP Victus 15-fa1xxx, Intel i7-13700H, integrated Iris Xe Graphics (other GPU was not used, Windows 11 (64 bit)) with the charger plugged in.

5.1 Experiment 1

During Experiment 1, a Naive Bayes model and a Random Forest model were compared on the inference and training speed. The Random Forest model first proved to be way slower, so it was reduced to just 5 decision trees, instead of the initial 100. This made it significantly faster, at the loss of little accuracy. The experiment was performed 500 times, to prevent fluke results and since the experiment was cheap to run. The mean and median accuracy for the Random Forest algorithm was 0.982. The mean and median F_1-score was also 0.982. The Naive Bayes model was correct less often, with a mean and median accuracy of 0.963, and a mean and median F_1-score of 0.962. The spread of these results is displayed in Figure 3. This improved performance of the Random Forest model comes at the cost of some training time, as is visible in Figure 4. (The middle line is the median, while the whiskers are placed at the 25_{th} and 75_{th} percentiles.) The Random Forest model takes longer to train, at around 0.05 seconds on average, while the Naive Bayes model

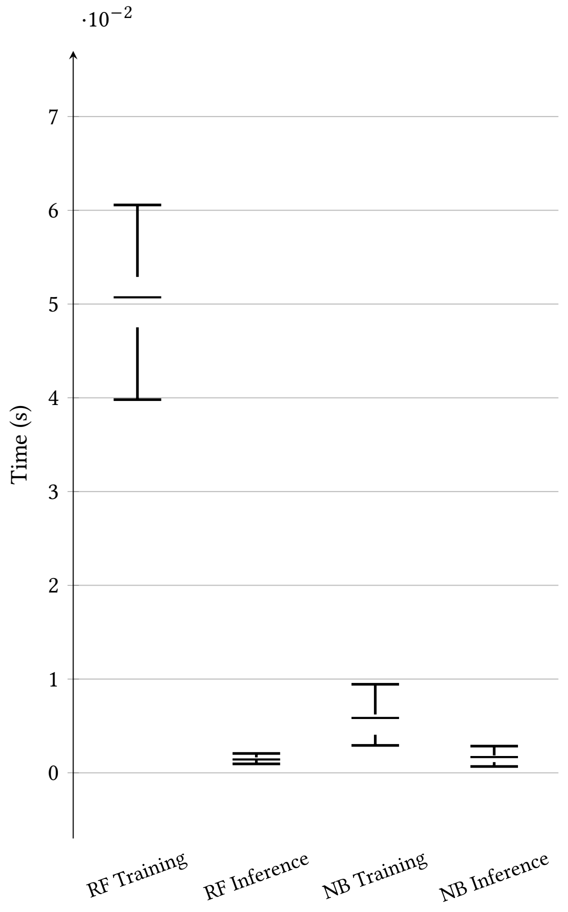

Fig. 4. Time usage

takes under 10 milliseconds. The inference times are very similar however with medians of 1.426 ms (Random Forest) and 1.681 ms (Naive Bayes).

5.1.1 Discussion. This experiment was deemed of lesser importance, so the comparison is not very thorough and can be expanded upon in many different ways. The data was taken from only one subject. Kamminga et al. highlights that this way of taking samples makes the model look better than it would perform in a realistic scenario. Using data from multiple subjects would make the model more able to classify samples from previously unseen subjects. It also unlocks a larger dataset. Using more efficient implementations (outside Python) of the models and tuning these to make optimal use of the system (the GPU and many CPU cores were not used during the experiment) to increase performance or decrease resource usage may be possible. During the experiment it was also found that many resource usage measurement libraries were limited or unavailable outside of Linux systems. Thus, a test on a Linux system could provide more insight into energy usage and the like. Moreover, Scalene was unable to measure the execution times in the experiment, as the functions were run too quickly. If the program could be slowed down (less powerful system, larger dataset, more features), this tool could be used to assess memory and processor usage. Using more and different datasets could also help with generalizing the conclusions. An LSTM was also not tested, so this forms a possibility to extend the experiment.

5.2 Experiment 2

For every strategy, the experiment was run 10 times, as it was not too costly in terms of time, while averaging multiple results prevents outliers from misrepresenting the usual performance. The batch size for the active learning strategy was 1, so the entire dataset was reconsidered after every sample. The active learning loop was stopped after the training set contained 1000 samples. The model that was trained on these 1000 samples was evaluated on the remaining samples The resulting average class distributions are displayed in Figures 5 and 6. Multiplying the values by 1000 returns the average absolute frequencies, as there are 1000 samples in the total training set each time.

Random sampling functions as a reference, as it preserves the relative distribution found in the dataset. It scores an H_{norm} of 0.586. The accuracy of 0.935 and the F_1-score of 0.931 are also used as reference. These values can be read from Table 3. MDS and MCS seem to provide no better distribution, and perform worse on the remaining samples. LCS and CES (Figure 5 however do affect the distribution significantly. On average, out of the 1000 samples selected, around 160 samples of the head class (Walking) are selected less. Moreover, these strategies manage to find more samples in the middle and tail classes. Very noticeable are the plusses in Running (±50-60%), Scratch-biting (±500%) and Head-shake (±900-1000%). This is particularly impressive given that the full dataset of 13838 samples contains 96 instances of Scratch-biting (±44% found) and just 49 of Head-shake (±77% found). The two algorithms also manage to find more samples of most rarest classes, such as Fighting, Rolling and Shaking. Their balance scores are 0.704 (CES) and 0.716 (LCS). This is accompanied by a higher accuracy and F_1-score to boot.

The regular density weighted (DW) approach looks to be of little use without the use of another strategy. The relative frequency in the head class (average frequency/1000) exceeds the bar plot's vertical limits at 0.9917. This approach It is not unexpected that the samples in the head class, being the most common, also resemble the collective of all samples more than other classes. The inverse density weighted approach (DW^{-1}) is more balanced however, and manages to very effectively avoid picking Walking samples, at just under 10%. It does have a tendency to gravitate towards the Grazing class, even more than CES and LCS did. Combining LCS with DW^{-1} was therefore a logical avenue to achieve an even more balanced class distribution. After trying out different β parameters, LCS DW^{-5} was found to create the most balanced class distribution, with an H_{norm} of 0.764.

The head classes are still not very balanced however, so PB may be able to help here. On its own, it does balance the common classes very well, but it is of little use finding tail class samples, as seen in Figure 6. LCS DW^{-5} combined with PB^{20} proves an improvement over LCS DW^{-5}. It even finds all 49 samples of Head-shake at some tests. It is probable that this class has a distinctive signature. The

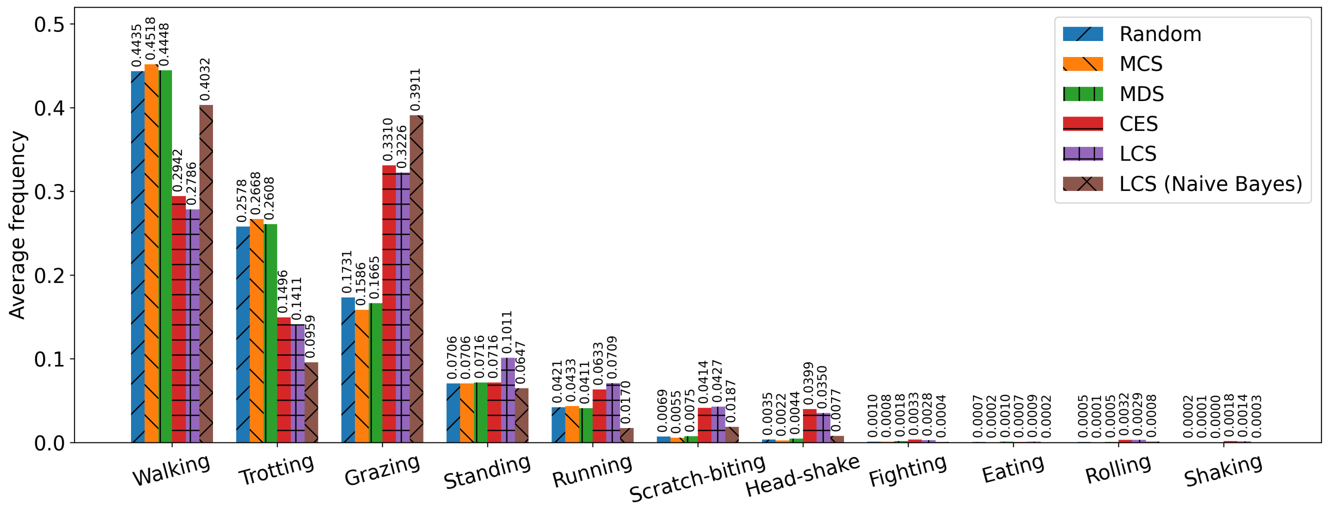

Fig. 5. Class Distributions - Base Strategies

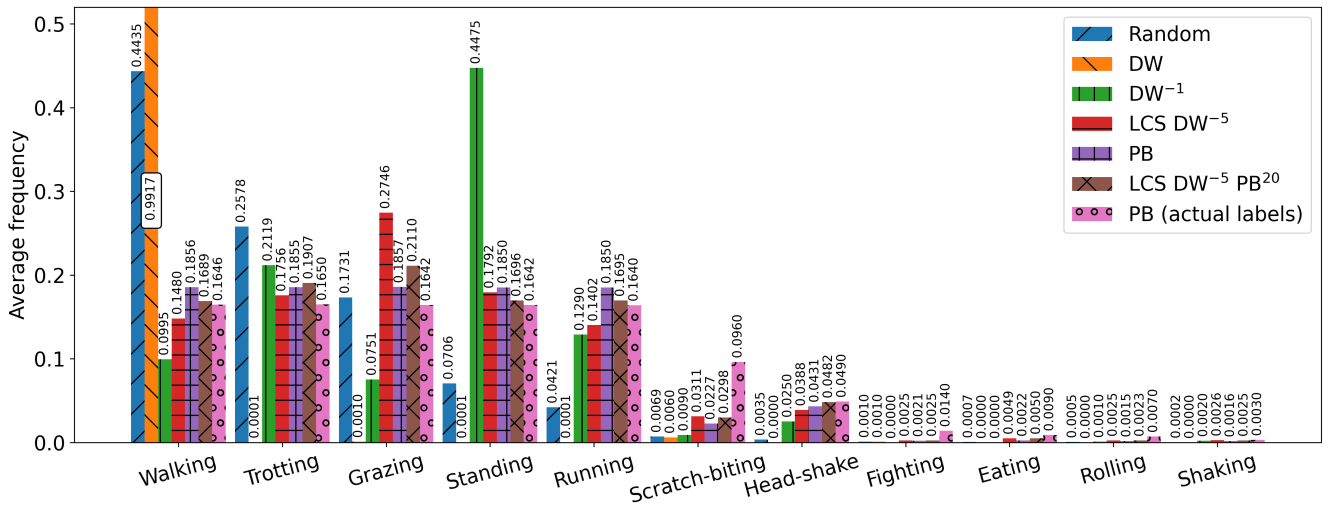

Fig. 6. Class Distributions - Extra Factors

H_{norm} sits at 0.778, nearly 0.2 points above random sampling. The PB method has also been run with knowledge of the actual labels, which results in finding all samples from the rare classes and increasing the H_{norm} to 0.839. This is the highest possible balance score with this dataset, as there are not enough samples for the rarer classes in the total sample set to create a uniformly balanced training set. This strategy is not realistic, but it is interesting that this 'balance' (high H_{norm}) does not contribute to a high performance, with accuracy and F_1-score below that of random sampling.

Out of curiosity, the effect of choosing the Naive Bayes model instead of the Random Forest model has also been briefly examined and the result is quite surprising; LCS with a Naive Bayes model does worse than random sampling (see Figure 5 and Table 3), on the balance score and on accuracy and F_1-score.

5.2.1 Discussion. The final result discussed calls for further investigation, but given the time constraints, the effect of using the Naive Bayes model with other strategies has not been tested. The best approach (LCS DW^{-5} PB^{20}) also could be tuned more finely, as only some integer values have been tested manually to find the optimal β and γ coefficients.

The correlation between performance and balance is not very clear in the results. The high balance of LCS and its variants corresponds with increased accuracy and F_1-score, but the extreme

Strategy	H_{norm}		Accuracy remaining samples		F_1-score remaining samples		Strategy
	Absolute	Relative	Absolute	Relative	Absolute	Relative	
Random	0.586	0	0.935	0	0.931	0	Random
MDS	0.575	-0.011	0.909	-0.026	0.914	-0.017	MDS
MCS	0.589	+0.003	0.933	-0.002	0.928	-0.003	MCS
CES	0.704	+0.118	0.953	+0.018	0.949	+0.018	CES
LCS	0.716	+0.130	0.963	+0.028	0.960	+0.029	LCS
LCS (NB)	0.554	-0.032	0.800	-0.135	0.753	-0.178	LCS (NB)
DW	0.023	-0.563	0.450	-0.485	0.318	-0.613	DW
DW^{-1}	0.638	+0.052	0.880	-0.055	0.880	-0.051	DW^{-1}
LCS DW^{-5}	0.764	+0.178	0.957	+0.022	0.953	+0.022	LCS DW^{-5}
PB	0.763	+0.177	0.944	+0.009	0.941	+0.010	PB
LCS DW^{-5} PB^{20}	0.779	+0.193	0.960	+0.025	0.957	+0.026	LCS DW^{-5} PB^{20}
PB (actual labels)	0.839	+0.253	0.894	-0.041	0.919	-0.012	PB (actual labels)

Table 3. Balance and performance scores

balance of PB with the actual labels decreases performance. It is likely that this is caused by the fact that performance is checked on the remaining data, which in that case consists of the 5 most common classes, while the model was trained on 11 classes. Given a more representative test set, which would strengthen any conclusions regarding performance, the performance may be different.

It can be questioned whether the uniform balance is best for model performance, as with LCS and CES (particularly LCS DW^{-5} PB^{20}), Head-shake samples are found often, while Scratch-biting and Fighting samples are not found as much, absolutely and relatively. It is therefore conceivable that more samples of the latter classes are needed for the model to recognize these classes reliably, while less are needed of the former.

A more rigorous statistical analysis would also be welcome, as some strategies deviate more in their resulting distributions than others. Some simplifications have also been made in this simulation. The oracle in this experiment is always correct for example. In reality, this is not always the case. [7] The oracle is also able to classify any 1 second sample in basically no time. To make the best use of a real oracle, it would not be ideal to have them label clips of 1 second at a time, because the time to switch between clips and other overhead would be too high. The results presented are also hard to compare to other studies, as most studies focus on performance statistics, which were not the primary objective in this research, and carry with them some methodological problems in this study, as discussed previously.

6 CONCLUSIONS

In this research, it has been demonstrated that a Random Forest model outperforms a Naive Bayes model in performance, while retaining similar inference times. Also shown is that active learning techniques Confidence Entropy Sampling and Lowest Confidence Sampling provide a more balanced class distribution while training a machine learning model. This is shown to result in an improvement in performance. Even more pronounced is their effect on the frequency of less common and rare classes in the training set, whose presence is increased greatly. The Inverse Density Weighted (DW^{-1}, 4.2.2) approach and the newly proposed Pragmatic Balance approach (4.2.3), which have also proven to be a fitting complement to these base active learning strategies, with the inverse Density Weighted score boosting the frequency of rare classes and the Pragmatic Balance score balancing head and less common classes. A combination of these complements and the LCS base strategy (LCS DW^{-5} PB^{20}) improves balance by 0.193, accuracy by 0.025 (2.5%.) and F_1-score by 0.026 (2.6%.). The class Rolling, which is important as a symptom of colics, was found over 4 times as often as with random sampling.

6.1 Research directions

After all has been said and done, some directions for future research can be identified, and will be briefly expounded below.

6.1.1 More extensive performance review. Experiment 1 was very brief. As mentioned, an LSTM was also planned to be tested, but due to to time constraints, this was abandoned. Comparing more models, perhaps with diverse datasets and on more resource-constrained (embedded) devices, would be interesting. It should then also be considered how models make use of memory and CPU/GPU.

6.1.2 Model versus active learning algorithms. The final result discussed in Experiment 2 opens a new rabbit hole. How do different active learning strategies combine with different model algorithms? It would be very lucky if the combination of Random Forest and Lowest Confidence Sampling is the best combination. Apart from algorithm choice, other dimensions can be considered, such as choice of model parameters (e.g. the amount of trees in Random Forest, amount of neurons in Neural Networks) and choice of batch size or number of active learning loop iterations.

6.1.3 Tuning of approaches. As mentioned in the discussion of Experiment 2, the chosen parameters for the additional factors (DW, 4.2.2 and PB, 4.2.3) have not been decided through a rigorous process. One could investigate more thoroughly the optimal parameters, and assess the construction and combination of these factors, as there may be a more mathematically sound or more effective way to devise these.

6.1.4 More data. As Kamminga et al. highlighted, using data of multiple subjects and perhaps even multiple species allows the model to be used more generally [14]. The presence of more data also makes some choices in this experiment infeasible, such as re-calibrating the score after every sample. Successfully using these approaches with a larger dataset and solving these scaling problems would therefore go a long way towards a real serious implementation.

ACKNOWLEDGMENTS

During the creation of this paper, generative AI (ChatGPT) was used to write some code for the LaTeX figures and graphs, as well as to improve some formulations. It has also been used to optimize some parts of the code for the experiments. The author has reviewed all work and takes full responsibility.

REFERENCES

[1] Emery D. Berger, Sam Stern, and Juan Altmayer Pizzorno. 2022. Triangulating Python Performance Issues with Scalene. https://doi.org/10.48550/arXiv.2212.07597 arXiv:2212.07597 [cs].
[2] Stephanie D. Bland. 2016. Equine colic: a review of the equine hindgut and colic. *Veterinary Science Development* 6, 1 (Aug. 2016). https://doi.org/10.4081/vsd.2016.6223 Number: 1.
[3] Alexander E.I Brownlee, Jason Adair, Saemundur O. Haraldsson, and John Jabbo. 2021. Exploring the Accuracy – Energy Trade-off in Machine Learning. In *2021 IEEE/ACM International Workshop on Genetic Improvement (GI)*. 11–18. https://doi.org/10.1109/GI52543.2021.00011
[4] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* 106 (Oct. 2018), 249–259. https://doi.org/10.1016/j.neunet.2018.07.011
[5] Cesar Castellon, Swapnoneel Roy, Patrick Kreidl, Ayan Dutta, and Ladislau Bölöni. 2021. Energy Efficient Merkle Trees for Blockchains. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 1093–1099. https://doi.org/10.1109/TrustCom53373.2021.00149 ISSN: 2324-9013.
[6] Pepijn de Reus, Ana Oprescu, and Koen van Elsen. 2023. Energy cost and machine learning accuracy impact of k-anonymisation and synthetic data techniques. In *2023 International Conference on ICT for Sustainability (ICT4S)*. 57–65. https://doi.org/10.1109/ICT4S58814.2023.00015 arXiv:2305.07116 [cs].
[7] U. Feige, A. Shamir, and M. Tennenholtz. 1990. The Noisy Oracle Problem. In *Advances in Cryptology — CRYPTO' 88*, Shafi Goldwasser (Ed.). Springer, New York, NY, 284–296. https://doi.org/10.1007/0-387-34799-2_22
[8] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. 2021. Recent Advances in Open Set Recognition: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 10 (Oct. 2021), 3614–3631. https://doi.org/10.1109/TPAMI.2020.2981604 arXiv:1811.08581 [cs, stat].
[9] Haibo He and Edwardo A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (Sept. 2009), 1263–1284. https://doi.org/10.1109/TKDE.2008.239 Conference Name: IEEE Transactions on Knowledge and Data Engineering.
[10] L. E. Hogeweg, R. Gangireddy, D. Brunink, V. J. Kalkman, L. Cornelissen, and J. W. Kamminga. 2024. COOD: Combined out-of-distribution detection using multiple measures for anomaly & novel class detection in large-scale hierarchical classification. https://doi.org/10.48550/arXiv.2403.06874 arXiv:2403.06874 [cs].
[11] Maartje Huveneers. 2021. The Development of a Horse Activity Recognition Algorithm. https://essay.utwente.nl/86921/ Publisher: University of Twente.
[12] Jacob W. Kamminga. 2019. Horsing Around – A Dataset Comprising Horse Movement. https://doi.org/10.4121/uuid:2e08745c-4178-4183-8551-f248c992cb14
[13] Jacob W. Kamminga. 2024. AI-Sensus/M-MOVE-IT. https://github.com/AI-Sensus/M-MOVE-IT original-date: 2022-09-07T13:34:33Z.
[14] Jacob Wilhelm Kamminga. 2020. Hiding in the Deep: Online Animal Activity Recognition using Motion Sensors and Machine Learning. (Sept. 2020). https://doi.org/10.3990/1.9789036550550
[15] Natasa Kleanthous, Abir Hussain, Wasiq Khan, Jenny Sneddon, and Alex Mason. 2020. Feature Extraction and Random Forest to Identify Sheep Behavior from Accelerometer Data. In *Intelligent Computing Methodologies*, De-Shuang Huang and Prashan Premaratne (Eds.). Springer International Publishing, Cham, 408–419. https://doi.org/10.1007/978-3-030-60796-8_35
[16] Zhao-Yang Liu and Sheng-Jun Huang. 2019. Active Sampling for Open-Set Classification without Initial Annotation. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (July 2019), 4416–4423. https://doi.org/10.1609/aaai.v33i01.33014416 Number: 01.
[17] Tong Luo, K. Kramer, S. Samson, A. Remsen, D.B. Goldgof, L.O. Hall, and T. Hopkins. 2004. Active learning to recognize multiple types of plankton. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 3. 478–481 Vol.3. https://doi.org/10.1109/ICPR.2004.1334570 ISSN: 1051-4651.
[18] Atefeh Mahdavi and Marco Carvalho. 2021. A Survey on Open Set Recognition. In *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. 37–44. https://doi.org/10.1109/AIKE52691.2021.00013 arXiv:2109.00893 [cs].
[19] Felix Nahrstedt, Mehdi Karmouche, Karolina Bargieł, Pouyeh Banijamali, Apoorva Nalini Pradeep Kumar, and Ivano Malavolta. 2024. An Empirical Study on the Energy Usage and Performance of Pandas and Polars Data Analysis Python Libraries. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE '24)*. Association for Computing Machinery, New York, NY, USA, 58–68. https://doi.org/10.1145/3661167.3661203
[20] Alan K. Outram, Natalie A. Stear, Robin Bendrey, Sandra Olsen, Alexei Kasparov, Victor Zaibert, Nick Thorpe, and Richard P. Evershed. 2009. The Earliest Horse Harnessing and Milking. *Science* 323, 5919 (March 2009), 1332–1335. https://doi.org/10.1126/science.1168594 Publisher: American Association for the Advancement of Science.
[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[22] Scikit-learn Documentation Team. 2024. f1_score. https://scikit-learn/stable/modules/generated/sklearn.metrics.f1_score.html
[23] Burr Settles. 2010. Active Learning Literature Survey}. *University of Wisconsin, Madison* 52 (July 2010).
[24] Burr Settles and Mark Craven. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Mirella Lapata and Hwee Tou Ng (Eds.). Association for Computational Linguistics, Honolulu, Hawaii, 1070–1079. https://aclanthology.org/D08-1112
[25] Claude Shannon. 2021. A Mathematical Theory of Communication (1948). (Feb. 2021). https://doi.org/10.7551/mitpress/12274.003.0014
[26] Ian F. Spellerberg and Peter J. Fedor. 2003. A tribute to Claude Shannon (1916–2001) and a plea for more rigorous use of species richness, species diversity and the 'Shannon–Wiener' Index. *Global Ecology and Biogeography* 12, 3 (2003), 177–179. https://doi.org/10.1046/j.1466-822X.2003.00015.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.1466-822X.2003.00015.x.
[27] Suzanne Spink, Jacob W. Kamminga, and Andreas Kamilaris. 2022. Improving the Annotation Efficiency for Animal Activity Recognition using Active Learning. In *Measuring Behavior 2022: 12th International Conference on Methods and Techniques in Behavioral Research, and 6th Seminar on Behavioral Methods*. 51–58. https://doi.org/10.6084/m9.figshare.20066849
[28] Aimee van Wynsberghe. 2021. Sustainable AI: AI for sustainability and the sustainability of AI. *AI and Ethics* 1, 3 (Aug. 2021), 213–218. https://doi.org/10.1007/s43681-021-00043-6

A SOURCE CODE

The source code for both experiments has been made available and can be checked at this link:

https://mega.nz/file/8qtSAAaD#PrL2xDMg96pYNC25pw0VioRYtXTvX6eRMaBHxgiqht0