UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

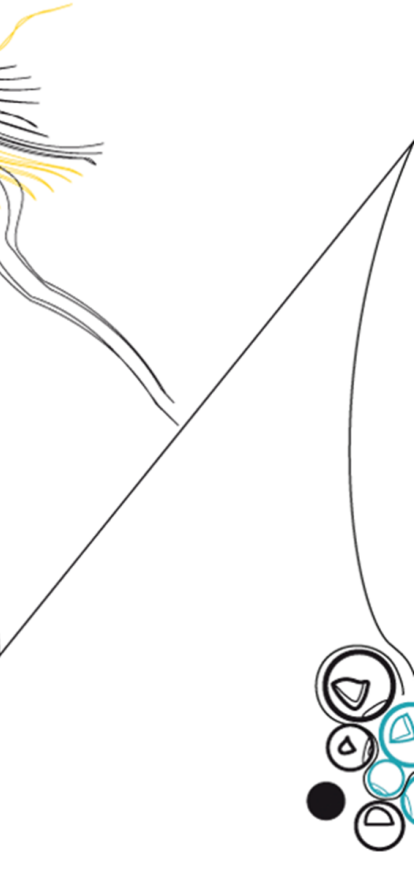
The Effective Use of Limited Feedback to Personalize a Mixed Model for Human Activity Recognition

Maartje Huveneers

Graduation Project

Interaction Technology

March 2024



Supervisors University of Twente
Mannes Poel
Gwenn Englebienne

Supervisor Nedap
Thomas Markus

Acknowledgements

I would like to start by thanking my supervisors. First, a massive thank you to Thomas Markus for all his feedback, support and critical thinking. Second, I want to thank Mannes Poel for his technical insights, feedback and patience with me during this project. Apart from guiding me in the process, you both gave me the necessary time and space during challenging periods.

Then, I want to thank all colleagues at Nedap, who gave me new ideas and skills, or whom I could simply just relax with. I am really glad that I met all of you. A special thanks to Steyn Potze, who supervised me during my Research Topics, gave me feedback on many pages of background research and always supported me. And thanks to Irene van der Zande, who joined me in my long journey from Enschede to Nedap and whom I could always spar with about my project.

Next, I want to thank the two data annotators for looking through many videos of skeletons for me. Lastly, thanks to my friends and family who supported me during this journey. I couldn't have done it without you.

Abstract

HAR (Human Activity Recognition) is an upcoming field of technology that can potentially decrease the workload of healthcare workers by recognizing abnormal behaviour. However, existing HAR systems often lack personalization, which can lead to reduced performance. This thesis proposes a new framework to personalize a mixed model, consisting of a general and personal model, using dynamic classifier selection and novelty detection. The personal model adapts to the subject over time by receiving feedback from annotators. With the goal of reducing annotators' efforts, LF (limited feedback) was applied, which allows annotators to select from a limited list of classes rather than the entire set, and Active Learning.

The experiments were conducted on three toy problem datasets and one HAR dataset. The results show that the framework performs similarly or better than the personal model in both datasets. When the personal model lacks performance, the framework can improve its performance, although it is occasionally outperformed by the general model.

Additionally, the comparison of five LF techniques (Correct Hard, Sampled Hard, Modified Soft, Unbiased Risk Estimator and Deep Naive Partial Label learning) showed that all techniques, except for Correct Hard, performed similarly to a model trained with full feedback. This indicates that the number of classes presented to the annotator could be reduced while yielding the same learning gain. However, all models suffered from catastrophic forgetting, which led to an overall low performance and could have reduced the difference in results. Next to that, entropy-based sampling was compared to random sampling but showed no improvement in reducing the number of samples required for optimal performance for both the toy problem and the HAR dataset.

While the methods were tested on two diverse datasets, future work should experiment with different datasets, subjects and models to see if the results generalize. Additionally, further research should explore different dynamic classifier selection and novelty detection methods to improve the framework. Lastly, other Active Learning strategies should be tested to determine whether they can reduce the number of samples required for optimal performance.

Contents

List of Figures

List of Tables

List of Symbols

C Comparison set, consisting of part of the feedback samples.

F Feedback set, consisting of all feedback samples.

I The set of indices corresponding to candidate labels..

P Periodic set, consisting of the most recent incoming samples.

T Training set, consisting of part of the feedback samples.

X The feature space.

Y The (soft) label as retrieved from the feedback of x_i.

$\hat{y}$ The predicted label of x_i.

a Active learning sample selection function.

b The index of the label from the full list labels.

c Binary model indicator (output of j, the dynamic classifier selection function).

d The distance function, which calculates the average distance to the k-nearest neighbours of a sample.

e The number of classes presented to the annotator[1].

g The general model prediction function.

k The number of neighbours of a sample.

m The number of classes.

n Binary novelty indicator (output of h, the novelty detection function).

o Model prediction (output of p or g, the personal/general models).

p The personal model prediction function.

s A set time period, after which feedback is requested.

t The current time period.

v The number of dimensions.

x_i Sample with index i.

y The actual label of x_i.

[1] Section 5.1.2 provides a more elaborate explanation on e.

List of Abbreviations

AL Active Learning. 3, 7, 16, 17, 19, 20, 23–25, 34, 35, 37–39, 41, 46, 48, 49, 59, 60, 62, 71, 72, 80, 84, 85, 87, 89, 90, 93, 94, 96–100

AP Average Precision. 61, 77

CLL Complementary Label Learning. 29, 32–34, 37, 46, 47, 91, 93, 96, 99

Conditional Prior Conditional Prior Soft. 30–32

Correct Correct Hard. 3, 7–10, 30, 31, 47, 69–71, 80–84, 90–93, 96, 97, 120, 124, 125, 127, 128

DCS Dynamic Classifier Selection. 7, 19, 22, 24–26, 28, 37–40, 42–44, 46, 59, 62, 64, 65, 87, 88, 90, 96, 99

DCW Dynamic Classifier Weighting. 22

DNPL Deep Naive Partial Label learning. 3, 8–10, 32, 47, 48, 69–71, 80–85, 90–93, 96–98, 120, 122, 124–128

EGL Expected Gradient Length. 36

EMC Expected Model Change. 36, 37

HAR Human Activity Recognition. 3, 9, 15, 17–19, 42–44, 46, 49, 50, 52, 53, 58, 60–62, 73, 74, 85, 87, 89–98, 124

k-NN k-Nearest Neighbours. 25–28, 37, 43, 44, 90

LCA Local Class Average. 25, 26, 28, 37, 44

LF limited feedback. 3, 8, 10, 16, 17, 19, 24, 25, 29, 32–34, 37, 42, 46–49, 57, 60, 62, 69, 71, 80–85, 87, 89–91, 93, 94, 96–100, 117, 120

MAD Median Absolute Deviation. 43

MCLs Multiple Complementary Labels. 33, 34

ML Machine Learning. 15, 16, 23, 46, 100

mmWave millimetre-wave. 15, 18

Modified Modified Soft. 3, 8–10, 30, 31, 47, 69–71, 80–84, 90–93, 96–98, 120, 121, 124, 125, 127, 128

MSP Maximum Softmax Probability. 26–28

ND Novelty Detection. 7, 19, 20, 22, 24–26, 28, 37–40, 42–44, 59, 62, 64, 65, 87, 90, 96, 99

OLA Overall Local Average. 25, 26, 28, 37

PL-BLC Partial Label learning with Batch Label Correction. 32

PLL Partial Label Learning. 29, 31, 32, 34, 37, 46–48, 91, 93, 96, 99

QBC Query-By-Committee. 36, 37

RQ Research Question. 18

Sampled Sampled Hard. 3, 7–10, 30, 31, 47, 69–71, 80–84, 90–92, 96–98, 120, 124, 125, 127, 128

ST-GCN++ Spatio-Temporal Graph Convolutional Network++. 57

SVDD Support Vector Domain Description. 33, 47

uCBLOF Unweighted Cluster-Based Local Outlier Factor. 26–28, 37

Uniform Uniform Soft. 30–32, 47, 48, 91, 93

URE Unbiased Risk Estimator. 3, 8–10, 33, 47, 48, 69–72, 74, 80–84, 90, 91, 93, 96–98, 118, 120, 123–125, 127, 128

1 Introduction

1.1 Motivation

The elderly population have a higher chance of injuries or illnesses, which in combination with the fact that the elderly population is increasing, indicates that more people require care. This causes an increase in the workload for healthcare workers. 51% of Dutch healthcare workers in nursing care have stated that their workload is too high [1].

Technological systems can be used to reduce the workload of healthcare workers. An example of such a technology is HAR (Human Activity Recognition), which detects the activities a person is performing. These activities can be used to recognize daily patterns and abnormalities [2]. [2]

Research is currently being performed using mmWave (millimetre-wave) radar technology for HAR [5], [6]. The main advantage of using mmWave is that it is not sensitive to lighting conditions and is less privacy-invasive than cameras because it does not capture characteristics identifiable by other humans. Therefore, this study aims to build upon mmWave systems. However, because of the limited availability of mmWave datasets, this study focuses instead on HAR datasets with a similar datatype as the mmWave can produce.

Studies have already shown the feasibility of HAR with a mmWave radar [6], although improvements are possible. One of the causes of mistakes in a HAR system is personal differences; everybody moves differently. A general assumption within many ML (Machine Learning) systems is that the full population acts similarly in a similar context, therefore yielding similar data. However, in many fields, among which HAR, people have individual differences. Due to these differences, two people might generate the same input, while leading to a different output.[3] This can lead to a decrease in performance if a generalized system is used [7]. Instead, a HAR system can gain information about the characteristics of an individual through feedback, which can increase its performance [8].[4]

[2] An example of this is that HAR can be used to detect falls [3] and nightly wandering [4].

[3] For example, when one person falls to the ground quickly, they might simply be doing sports, while similar movements for another person may indicate a fall.

[4] Feedback can not only improve the performance of a ML system, but also increase the acceptance of users in it. Even though ML systems outperform human predictions in many tasks, people are often hesitant to use it when they discover that the system is imperfect. However, when allowing users to give feedback to the system, they are more likely to use it [9].

1.2 Problem statement

When incorporating feedback into a system, many studies ask annotators to choose the correct class out of the full list of classes [10]–[13].[5] However, labelling samples can be quite time-consuming, especially when there are many classes possible. While feedback could be gathered from multiple annotators, i.e. patients, healthcare workers and caregivers, it is unlikely that annotators will be open to finding the right label in a long list of classes for every feedback sample. Instead, annotators could be asked whether the correct activity is listed in a number of options. This study defines such feedback as LF (limited feedback), which can result in a hard or soft label, depending on the response. If one of the presented options is correct, the feedback can directly be used as a labelled sample for training the model, thus leading to a hard label. However, if none of the presented options is correct, no hard label can be retrieved. Instead, this limited information shows which classes are not the true label, and thus also, a list of classes among which the correct label is, i.e. a soft label. Unfortunately, soft labels cannot be used directly as input into most ML systems. Therefore, research is needed on how to handle soft labels in an ML system and how many options are optimal in terms of minimizing the time required to answer a question while maximizing the learning gain of the model.

When minimizing the work of giving feedback for annotators, not only the time per question should be minimized, but also the number of questions. AL (Active Learning) aims to use feedback more effectively by measuring the information gain for each sample. This allows annotators to answer fewer questions to gain the same performance level.

After gathering the feedback, it can be applied for personalization in a number of different ways, among which a mixed model [10], [11], [18], personalized model [18]–[20][6], transfer learning [7], [13], [17], [21]–[24] and reinforcement learning [25]–[28]. A mixed model combines two classifiers, a general and personal model, into one prediction (see Chapter 2.1 for an extensive explanation). This study uses a mixed model, rather than one of the other methods, because it can operate without any feedback[7], it is transparent in terms of which classifier made the final decision, and it can rely on the general model if the personal model

[5] Other methods for giving feedback are also possible, but have their disadvantages. For example, an annotator could be asked the relatively open question: *"What do you think should be the output of the system?"*. Depending on the application, this process may be too time-consuming for the annotator. Additionally, according to Cohn et al., "it is easier to criticise than to construct" [14, p. 17]. Therefore, they recommend proposing a solution or prediction to the annotator and asking for improvements, rather than asking the annotator to provide an answer themselves. Apart from requesting feedback on the output of a model, one can also ask for feedback on the features, rules or parameters of a ML model directly [15]–[17]. The downside of these types of methods is that the annotator needs to understand the explanation to some extent. If the annotator is unable to understand this, they are also unable to provide helpful feedback. Therefore, this study asks annotators to select an activity from a list of activities.

[6] Please note the personalized model is different from the personal model referred to later in this study.

[7] In the situation where no feedback is gathered yet, the mixed model can simply rely on the predictions of the general model.

fails to train well. This study proposes a new framework, based on the mixed model, to decide the best classifier to use based on the incoming test sample.

1.2.1 Research Question

As explained above, this study aims to investigate effective ways to use LF for personalization, leading to the following research question:

RQ *How can limited feedback be used effectively to personalize a mixed model in the context of HAR?*

To answer this question, the following sub-questions have been identified:

SQ1 *What is the trade-off between the maximization of learning gain and the minimization of the level of detail of feedback?*

SQ2 *How does the effective use of limited feedback determine the minimal number of feedback samples required to achieve an optimal level of performance?*

This study has two main requirements, which are described by the two sub-questions. First, the system should not require extensive answering to minimize the workload of annotators. SQ1 describes the trade-off that arises from this, where the system should maximize its learning gain, but minimize the annotators' workload. Second, the system should be able to adapt quickly, so that less feedback is required to personalize it, thereby further reducing the load for annotators. To use feedback effectively, AL will be used, a commonly used technique, especially when using neural networks. SQ2 explores the minimal number of feedback samples when using the feedback effectively through AL.

1.3 Scope

Since the fields of HAR, personalization and giving feedback each introduce their own additional challenges, this subsection will describe which challenges will be tackled in this study and which will not. Table 1 shows a summary of the related challenges which will be described shortly below.

Field	In scope	Out of scope
HAR	Classification problem	Highly imbalanced classes
	Handle complex data	Multiple labels per sample
	Handle multiple classes	Multiple subjects
Feedback	Work without any or with few feedback samples	Weak labels
		User interface
		Real-time
Personalization	Newly emerging input	Newly emerging classes
	Known input, with a different class Drifted input	

Table 1: Related challenges which are in and out of scope for this study.

1.3.1 In Scope

The RQ (Research Question) will be answered in the context of HAR. Since HAR is a classification problem, the RQ will only be considered in terms of classification and not regression. This choice was made to keep the problem simple, instead of increasing the complexity by allowing regression tasks.

MmWave data is converted to skeletons, which contain many complex features. As an indication, HAR datasets generally include 3D information about 10 to 30 joints for multiple frames [29]–[31]. Moreover, HAR can concern itself with many different classes, often at least eight, but can extend to many more [29]–[31]. Therefore, the solution in this study should also be able to handle at least multiple different classes. When deploying the system it will not have received any feedback samples yet, but it should still perform sufficiently then. Moreover, when starting to receive samples, the model must adapt quickly so that not too many samples are required to adapt the model. Lastly, there are multiple types of personalization possible, from which this study only examines a few: a newly emerging input in a previously unexplored area of the input space, a known input which maps to a different class and drifted inputs.[8]

1.3.2 Out of Scope

There are multiple challenges within HAR, giving feedback and personalization that will not be tackled to the scope of the study. HAR data is often highly imbalanced since some activities are generally performed more often than others [32]. This study focuses on balanced datasets so that the effectiveness of the feedback incorporation methods can be measured

[8] See Section 5.2.1 for more information on how these types of personalization are applied and why they might occur.

solely in terms of adaptability, rather than focusing on the problem of imbalance. While multiple labels per sample is possible within HAR, this study only considers single-labelled activities. Additionally, HAR also includes interactions between multiple people, but this study focuses only on actions involving one person.

Giving feedback also brings challenges, such as weak labels. When annotators label activities, it is reasonable to expect them to sometimes select an incorrect label. However, this study assumes that all given labels are correct to scope the problem. Additionally, this study focuses on the gathering and processing of feedback from a technical viewpoint, therefore the design of a user interface for giving feedback is not discussed. Lastly, the goal of such a system is to use it in real time. However, this study simply requires it to be efficient enough to run the experiments, but not necessarily that it runs in real-time.

Apart from the types of personalization mentioned earlier, other types will not be explored in this study, among which is the problem of newly emerging classes.[9]

1.4 Structure

Chapter 2 describes the general structure of a mixed model, its components DCS (Dynamic Classifier Selection) and ND (Novelty Detection), and introduces the topic of AL. Chapter 3 continues with the related work on these topics, along with related work on LF. After the components of ND and DCS have been explored, Chapter 4 describes the design of a novel framework. Next to that, Chapter 5 describes the choices made in terms of the methodology of the study, after which the results of the study are shared in Chapter 6. Furthermore, a discussion of the results and the limitations of the study are discussed in Chapter 7. Lastly, Chapter 8 concludes the thesis by revisiting the research question and discussing future work.

[9]There are multiple solutions to this in the field of Open Set Recognition [33], [34], although these would overcomplicate the problem in combination with personalization and LF. Even though the closed set assumption is not true within HAR, unknown classes will have a significantly lower frequency than known classes, since they have not appeared during training [35]. Therefore, the impact of unknown classes may be limited, and the choice was made to leave the problem of newly emerging classes out of scope.

2 Background

Since a mixed model will be used in this study, a definition of its structure is required to understand the following steps. Therefore, this chapter explains the definition of a mixed model and its components. The mixed model involves a combiner function and ND (Novelty Detection), which will be defined and elaborated on below. Additionally, the field of AL (Active Learning) will be explored to use feedback effectively. Therefore, AL and its subtypes will be explained in this chapter.

2.1 Mixed Model Structure

As described in Chapter 1, this study will use a mixed model for personalization through feedback. Mixed models consist of a general and a personal model that are combined with a combiner function to yield one prediction per sample.

A mixed model combines two separate models: a general model and a subject-dependent model [18]. General models train on data from a big population. The assumption is that, if the population is big enough, such a model will generalize towards new subjects. Therefore, it is assumed that the specific subject context is irrelevant to the predictions of a model [18]. However, in reality, this is often not true, since multiple studies [7], [18], [24] have shown a decreased performance for generalized learning, as opposed to personalized learning.

In contrast to a general model, a subject-dependent model learns solely from the data of a specific subject. Consequently, training a subject-dependent model requires sufficient labelled data from each subject, which is a downside, since data labelling can be a time-consuming and expensive process.

A mixed model combines a general model and subject-dependent model by training (at least) one model on data from many different subjects and training (at least) one model on data from the specific subject [18]. The goal of a mixed model is to combine the advantages of both subject-specific and general models. General models might perform quite well without any subject-specific data, while subject-dependent models might improve the performance in certain classes where the subject performs differently from the general population. Figure 1 shows the pipeline for a mixed model. Please note that the block "combine models" consists of a combiner, which either selects a model or weights the predictions of all models [36].

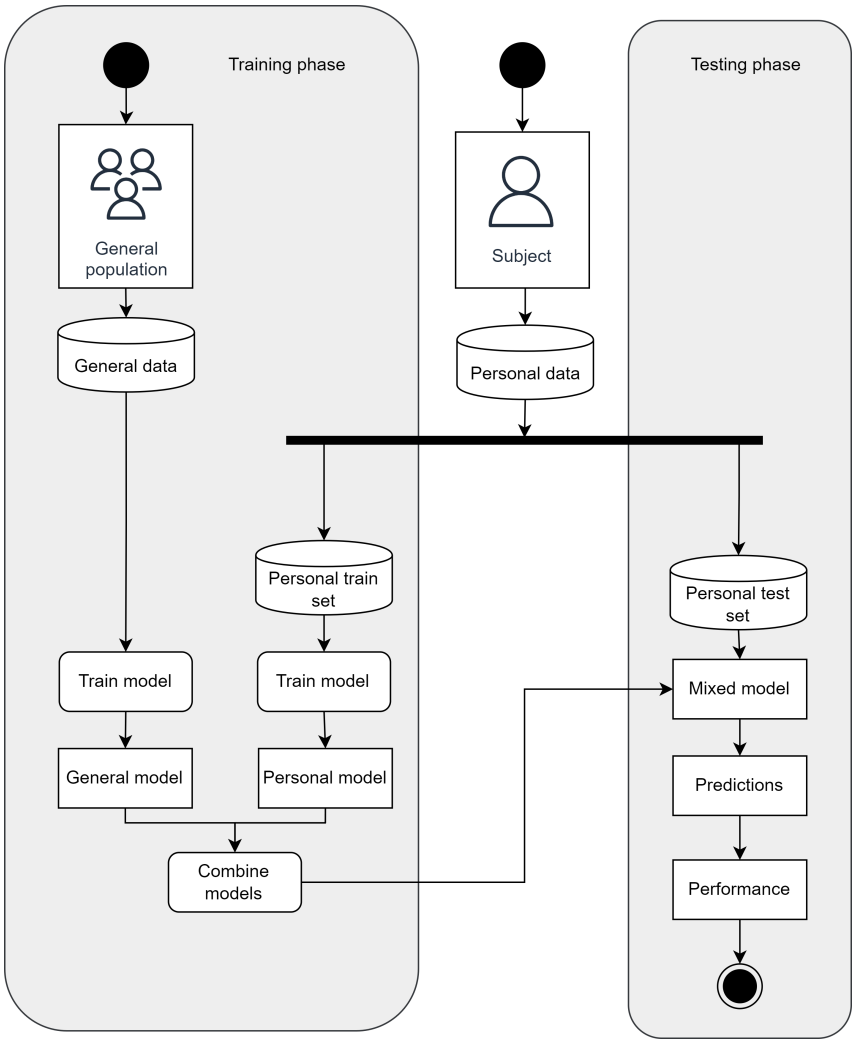


Figure 1: Pipeline of a mixed model, consisting of a general model, personal model and a combiner function.

2.2 Combiner Functions

A combiner function can be either dynamic or static. A static combiner selects or weights the classifiers regardless of the characteristics of a sample [37]–[39]. Examples of static combiners are majority voting, adding the a posteriori probabilities of classes or a trained weighting function [36], [37]. The disadvantage of static combiners is that they assume that the classifiers perform equally well for all samples. However, in this study, the general model may be better at classifying certain samples than the personal model and vice versa. In these situations, a static combiner may not perform well enough.

A dynamic combiner, on the other hand, takes information about the sample into account and can adjust its approach according to the incoming sample [36], [40]–[45]. Especially when a personal model has trained more on certain classes than others, it can be beneficial to

select/weight the classifiers based on the characteristics of a sample. Therefore, this section will only discuss dynamic combiner methods.

Dynamic combiners can perform either DCS (Dynamic Classifier Selection) or DCW (Dynamic Classifier Weighting). DCS aims to select the best-fitting classifier for a sample, whereas DCW weights the classifiers' output based on their estimated expertise about a sample.

The advantage of DCW is that it takes both classifiers into account when they are deemed equally useful. DCS, on the other hand, requires the choice of one classifier, even when they are equally useful. However, the advantage of DCS is that it is more transparent than DCW because it can be indicated exactly which classifier was responsible for the prediction. Transparency is a valued quality in healthcare systems because it allows patients, healthcare workers and developers to understand the system more, which can, in turn, improve the system and increase their trust in the system. Therefore, only DCS methods will be considered in this study.

When using DCS, the combiner function must have enough information about the input space. Otherwise, the risk of choosing the wrong model for the test sample increases. To counter this problem, it is possible to first test whether a sample is novel to the dataset and only perform DCS if the sample is not novel. The next section will introduce the concept of ND (Novelty Detection)

2.3 Novelty Detection

ND aims to detect test samples which do not fall into the distribution of the training samples it has encountered thus far. Similar fields include one-class classification, out-of-distribution detection, anomaly detection and outlier detection [46], [47]. While there are subtle differences in the goals of these fields, their approaches are generally similar and can be used across the fields.

In this study, samples are considered novel if no feedback has been given on similar samples. Thus, if a type of sample has been encountered many times in the general dataset but never in the personal dataset, it is considered novel. On the other hand, if it was never encountered in the general dataset, but has been encountered multiple times in the feedback set, it is not considered novel.

ND can be performed locally and globally, where samples are considered to be locally novel samples when they are novel in comparison to their close neighbourhood, while globally novel samples are novel in comparison to the full input space [46]. In this study, it is assumed that DCS can still make a reasonable indication of performance on locally novel samples, thus not requiring ND. Instead, ND will only be required to detect globally novel samples. Therefore, only global novelty detection is considered in this study.

Furthermore, three types of samples can be considered novelty: point, contextual and collective novelties [48]. Point novelties consist of one sample which stands out from the rest. Contextual novelties also consist of one sample but they only stand out in a specific context.

For example, a temperature of 25 degrees Celsius is not a novel sample, but in the context of winter, it is a novel sample. Lastly, collective novelties consist of a group of samples that stand out from the rest. On their own, they are not considered a novelty, but all samples together do form a novelty. In this study, only point novelty samples are relevant because the aim is to find individual samples that stand out from the other samples without contextual information.

2.4 Active Learning

Gathering labels for unlabelled samples can be a very time-consuming process, especially in a system that continuously gathers data. If it is only feasible to label part of the samples, careful consideration should be used in selecting these samples.

AL (Active Learning) is the sub-field of ML (Machine Learning) which concerns itself with this challenge: choosing which samples should be labelled first [49]. Figure 2 shows an example of sampling using AL through uncertainty sampling, as will be explained in Section 3.3.2, and random sampling. Careful selection of the samples allows a classifier to better learn a decision boundary with few labelled samples. This shows that AL can help to increase the performance while keeping the number of labelled samples fixed.

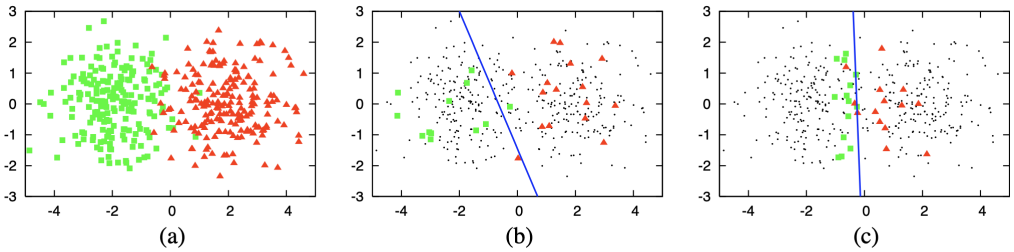


Figure 2: An example of sample selection through uncertainty sampling (an AL technique) and random sampling. (a) shows an example of a dataset, with samples belonging to a green or red class. (b) and (c) each shows 30 labelled samples in colour. (b) illustrates random sampling, while (c) illustrates uncertainty sampling. Source: [50]

There are three types of AL: stream-based AL, pool-based AL and query-synthesis methods [49]. Stream-based AL receives data in a stream, where new data is arriving continuously. The system then decides, based on the informativeness of the sample, whether it should be labelled or not. Pool-based AL, on the other hand, receives a set of data, which is fixed from the beginning. Therefore, pool-based AL ranks all samples in a pool of data based on informativeness, from which it queries the highest-ranking sample(s). Pool-based AL techniques can also be applied to stream-based data by collecting data samples from a longer period, e.g. all samples of one day. Lastly, query-synthesis methods generate data samples that would be the most informative data points [51]–[54]. This is especially useful when the number of samples is small. Importantly, this method can only be used when

synthesized queries are relevant. For example, in digit recognition, a non-existent digit is not relevant to query. Since this study is specifically concerned with giving feedback, synthesized samples are not relevant. Therefore, only stream-based and pool-based AL methods will be considered.

Additionally, each of these types of AL can be applied in serial or in batches [49]. Serial-mode AL asks the annotator for a label one sample at a time. This allows a classifier to update itself according to the received label and ask for a new sample which is most useful for the updated classifier. On the other hand, batch-mode AL asks the annotator to label multiple samples at a time. In this case, it receives no updates about the labels until all samples have been labelled. Because of this property, batch-mode AL requires diversity in its sample selection, to not select only similar samples. Research has shown that regular AL techniques generally perform poorly in batch mode, even worse than random sampling, because they tend to select many similar samples in batch [49]. Specific batch-mode AL techniques have been proposed [55]–[57], although they increase the complexity of the system. Since a simpler method is preferred in this study, only serial-mode techniques will be considered in this study.

The next chapter continues with an overview of related work into the topics discussed in this chapter, namely DCS and ND methods, followed by related work into LF (limited feedback) and AL (Active Learning).

3 Related Work

This section will describe the related work in the following four topics: DCS (Dynamic Classifier Selection), ND (Novelty Detection), LF (limited feedback) and AL (Active Learning). The related work in DCS and ND will help construct a novel framework. Moreover, related work in LF and AL will help answer the sub-questions.

3.1 Mixed Model

3.1.1 Dynamic Classifier Selection

As explained in Chapter 2.2, DCS (Dynamic Classifier Selection) is recommended for this system. While many DCS methods exist [36], [40], [42]–[45], this section will focus on the following DCS methods: a trainable combiner per classifier, k-NN (k-Nearest Neighbours) with OLA (Overall Local Average) and LCA (Local Class Average) and decision space with OLA.

Xu et al. [41] propose a DCS combiner consisting of one trainable combiner per classifier. In this scheme, the trainable combiner decides whether each classifier is an expert on a certain part of the input space. The trainable combiner consists of a neural network with one hidden layer, which uses the sample as input and outputs a score. If the score is higher than a threshold, the trainable combiner will be considered an expert. The trainable combiner is taught by inputting samples, along with the label *expert* or *no expert*, depending on whether the classifier classified the sample correctly. If one of the classifiers is selected as an expert, their classification is taken as the classification of the system. If both classifiers are considered experts, one of their classifications is selected. If none of them is considered an expert, the sample is rejected by the system. This means that the system indicates it is unable to classify the sample. The advantage of trainable combiners in general is that they require little manual tuning of parameters. However, such a network can be very complicated, since it has to model the behaviour of both classifiers to a certain extent. Training such a complex network with few samples could therefore easily lead to overfitting.

Woods et al. [58] propose DCS-LA, which does not require a complex network. It consists of two steps: to define a region of relevance and to select a classifier based on a selection criterion. The region of relevance defines which samples are similar to the test sample. Woods et al. select these samples using a k-NN (k-Nearest Neighbours), which finds the k samples which are the closest to the test sample according to a distance function. In this study, the Euclidean distance was used with normalized features. After finding the region of relevance, the classifiers are compared by a selection criterion on the region of relevance. Woods et al. compared two selection criteria: OLA (Overall Local Average) and LCA (Local Class Average). OLA simply calculates the accuracy of all samples in the region of relevance, while LCA only calculates the accuracy for the samples predicted as the same class as the

test sample. Woods et al. found that LCA outperformed OLA for multiple datasets. Other often-used selection criteria are based on ranking, confidence estimation and behaviour [36].

A downside to using a k-NN is that the performance of the method relies highly on the choice of k, while this is often a relatively subjective choice [59]. When k is chosen too small, noisy neighbours might be too influential, while if k is chosen to be too big it would evaluate the global performance of the classifiers rather than the local performance. Another downside is that using a high-dimensional dataset in combination with a k-NN causes two issues: computational power and diminishing distances. The problem of computational power can be solved by using more efficient algorithms such as a tree structure or an approximate k-NN [60]. Next to that, the distance measure can be chosen effectively to increase efficiency, for example by using the cosine distance metric over the Euclidean distance [61]. Next to that, the contrast between near and far neighbours diminishes when using high-dimensional data [62]. The main reason for this is that high-dimensional data generally also introduces many irrelevant dimensions, which influence the distances of all samples, therefore causing them to become similar. Even though these disadvantages are known, k-NN is the most often-used method for DCS [36].

A different method for finding the region of relevance is to find similar samples based on the decision space, i.e. the output, rather than the input, of a test sample. The output of the classifiers can consist of the predicted hard labels or the estimated posterior probabilities. Giacinto and Roli [42] propose using a similarity function which takes both classifiers into account. They loop through all training samples and check whether the classifiers predicted the same class for this training sample as for the test sample. If both classifiers predict the two samples as the same class, it is added to the region of relevance. Please note that this does not mean that both classifiers agree on the predicted class, it simply means that they individually classify the two samples as the same class. For example, if the general model classifies the test and train samples as *Run* and the personal model classifies the test and train samples as *Walk*, this sample is added to the region of relevance, while the two classifiers disagree. After finding the region of relevance, Giancinto and Roli choose the classifier with the highest OLA.

3.1.2 Novelty Detection

As mentioned in Chapter 2.3, ND (Novelty Detection) is an essential part of the framework used in this study. Four global point novelty detection methods will be discussed in this section: a k-NN with a threshold, a k-NN with a comparison of the distance to the neighbours' neighbours, uCBLOF and MSP. Please note that this is, however, not an exhaustive review of ND methods since many more methods exist [48], [63]–[69].

Many techniques have been proposed to achieve ND, which often focus on density-based measurements [46]. For example, Lazarevic et al. [70] use a k-NN to measure the distance between the test sample and its nearest neighbours according to some distance measure. Intuitively, samples with a high distance from other samples are more likely to be novel.

They compare the distance between the test sample and their nearest neighbour with k=1 to a certain threshold. All test samples that yield a higher distance to their nearest neighbour than the threshold are considered novel. The threshold is based on the training data, where the distances from all training samples to their neighbours are sorted. The threshold is chosen so that 2% of the data is considered novel. The disadvantage of this method is that the percentage of novel samples needs to be known or approximated.

Another way of using the distances from a k-NN is described by Ding et al. [71]. If $k = 1$, they compare the distance between a test sample and its nearest neighbour, d_1, to the distance between the neighbour and their nearest neighbour, d_2. If $k > 1$, d_2 is defined as the average distance of the nearest neighbour to their k nearest neighbours. Figure 3 describes the idea behind this approach. The intuition is that if $d_1 > d_2$, the neighbour is in a more densely populated area than the test sample, therefore this sample is classified as *novel*. On the other hand, if $d_1 < d_2$, the test sample is in a more densely populated area than its neighbour and therefore classified as *not novel*.

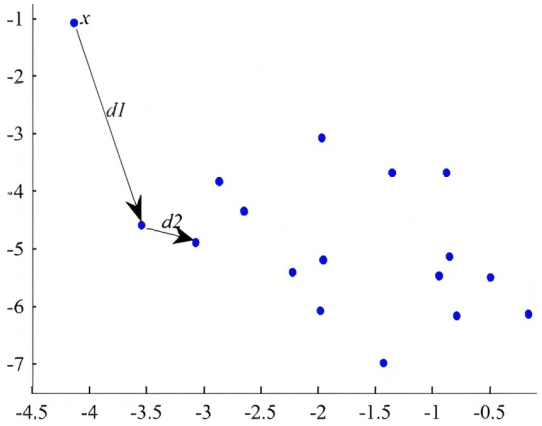


Figure 3: The distance comparison between d_1, a test sample x and its nearest neighbour and d_2, the nearest neighbour and their nearest neighbour. This figure assumes k=1, where it compares only the distance to its first nearest neighbour. For a higher value of k, the distance to the k nearest neighbours is averaged. Source: [71].

Apart from measuring density based on distance, multiple methods use clustering algorithms to measure the density around a test sample. Goldstein and Uchida [46] propose uCBLOF (Unweighted Cluster-Based Local Outlier Factor). First, it clusters the training data through k-means clustering. After defining the clusters, the distance is measured between the test sample and the centre of the cluster it was assigned to. If this distance exceeds a predefined threshold, the sample is considered to be novel. Choosing k for the k-means clustering algorithm can be difficult since the algorithm is sensitive to the choice in k [46]. There are possible solutions to this, such as applying k-means many times and choosing the most stable result overall or using a silhouette score [72].

Multiple studies also base their detection algorithm on the output of a classifier. For example, Hendrycks and Gimpel [73] compare the MSP (Maximum Softmax Probability)

of a model on a test sample to a threshold to detect novel samples. The idea behind this is that a classifier is often more uncertain about novel samples than known samples. However, as Hendrycks and Gimpel mention as well, the softmax value is often over-confident, even when an incoming sample is significantly different from the training data [74]. While they argue that the differences between novel and known samples are big enough to use for ND, this may not be the case in all situations. For example, in this study, the input of feedback samples may be similar to general training data, while the output should be different. A general model would most probably be overconfident in such cases, classifying it as a known sample, even though the sample, in a personalized context, has not been encountered before. Therefore, it is unclear whether MSP would work well in a personalization context.

3.1.3 Conclusion

This section has described related work into DCS and ND, which will be used to define a framework. Three DCS methods were described: a trainable combiner per classifier, k-NN with OLA or LCA and decision space with OLA. The disadvantage of using a trainable combiner per classifier is that it requires an additional training phase. The k-NN with OLA or LCA and decision space with OLA are relatively simple while yielding good results and thus seem like promising methods to use in this study. However, the decision space can lead to wrong results because neural networks can classify samples overconfidently incorrect [74]. Instead, using the similarity in input may result in a more relevant selection. Therefore, a k-NN is the recommended method to perform DCS in this study. Furthermore, the choice was made to evaluate the models with LCA than with OLA, since LCA generally yields better performance estimates than OLA [58].

Additionally, ND can be used to detect whether DCS has enough information to choose a classifier reliably. Four ND techniques were discussed: a k-NN with a threshold, a k-NN with a comparison of the distance to the neighbours' neighbours, uCBLOF and MSP. A disadvantage of MSP is that it is often not a good indicator of the novelty of samples. The other three methods seem promising to use in this study. The recommended method for this study is a k-NN since it has shown to have satisfactory performance on multiple anomaly detection datasets [71], [75], [76] and is relatively easy to implement and visualize. An additional advantage to this is that the same k-NN can be used for ND and DCS, such that the nearest neighbours only have to be found once and the system can perform more efficiently.

While most DCS and ND studies assume that a specific label can be retrieved for every sample, this is not the case in this study. The next section will discuss how to incorporate this limited feedback.

3.2 Learning from Limited Feedback

Gathering feedback on a long list of classes can be quite time-consuming. Therefore, it would be beneficial to ask an annotator for feedback on just a few options of classes. This type of

feedback is defined as LF (limited feedback) in this study. However, when the correct class is not among the proposed classes, this results in a list of candidate classes. While it is known that one of these classes is the correct class, one hard label cannot be given. This section will describe methods which deal with this situation; namely that multiple (but not all) labels are candidates for the real label.

Three promising fields have been identified that aim to extract the most information from limited feedback[10]: Probability Distribution, PLL (Partial Label Learning) and CLL (Complementary Label Learning). While several other studies propose solutions to learn from LF, many are not directly applicable to this study. For example, approaches that use one binary classifier per class [77] become impractical as the number of classes increases. Similarly, reinforcement learning techniques [27], [78] generally require a lot of feedback samples. Next to that, certain studies require other model types such as the maximum a posteriori [24] or are designed for different contexts, like the Markov decision process [28] and are therefore not recommended. Therefore, these approaches are not further explored. Instead, this section will focus on implementations using a probability distribution, PLL and CLL.

3.2.1 Probability Distribution

Lucas et al. [79] explore this problem specifically for binary feedback, as shown in Figure 4.

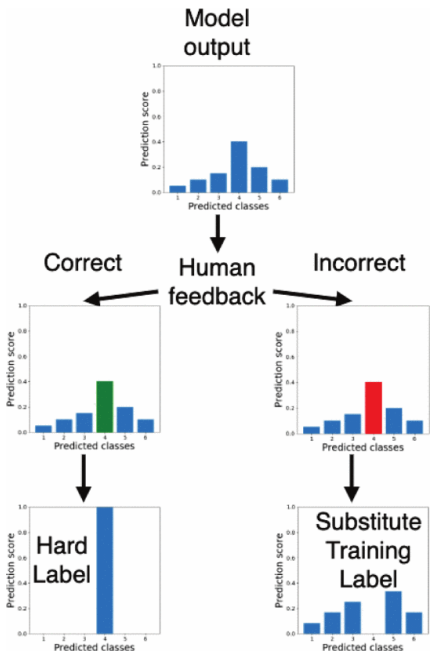
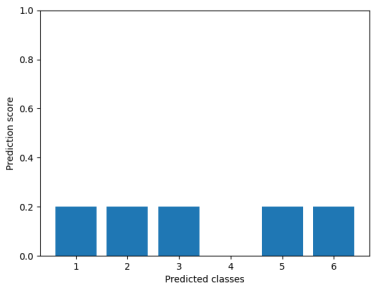


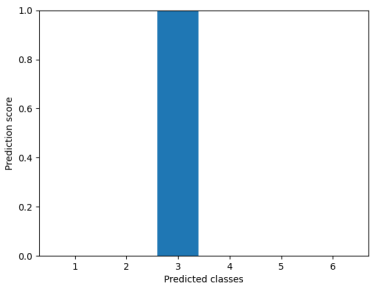
Figure 4: The pipeline for giving binary feedback on a certain class, either resulting in a hard or substitute label. Source: [79]

[10]The studies are not necessarily designed for handling LF, but they can be used in this situation since they handle the same input types.

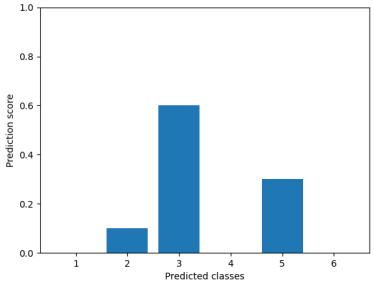
They proposed five methods, which are depicted in Figure 5, and consist of *Uniform (Uniform Soft)*, *Correct (Correct Hard)*, *Sampled (Sampled Hard)*, *Conditional Prior (Conditional Prior Soft)* and *Modified (Modified Soft)*.



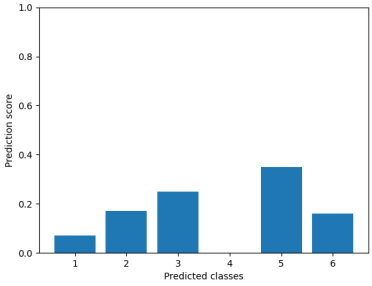
(a) *Uniform*



(b) *Correct* / *Sampled*



(c) *Conditional Prior*



(d) *Modified*

Figure 5: The different methods used by Lucas et al. to represent labels from binary feedback. Please note that both the Correct and Sampled techniques yield one hard label and are therefore both depicted in (b). Adapted from: [79]

First, the *Uniform* labelling technique uses a uniform distribution for all candidate labels. Thus, all candidate labels will get the same probability, regardless of any information gained by the model. Only the confirmed incorrect label yields a different probability, namely 0. Second, the *Correct* labelling technique simply disregards any samples that received negative feedback and only considers the positive feedback. Third, the *Sampled* labelling technique randomly samples one label out of the candidate labels. All candidate labels have an equal chance of being sampled. This does mean that the sampled hard label can consist of an incorrect label. Fourth, the *Conditional Prior* labelling technique estimates the conditional probabilities of all candidate labels using a held-out dataset. Last, the *Modified* labelling technique uses the output of the model as a soft label, with the output prediction of the incorrect class being set to 0.

They tested their approaches on four simple datasets; of which three consist of (5, 10 and 15) evenly spaced clusters and one dataset consists of 5 clusters with random centers. Moreover, they also tested their approaches on two complex datasets, namely MNIST and CIFAR10.

The *Uniform* labelling technique outperforms all other (*Correct*, *Conditional Prior*, *Sampled* and *Modified*) techniques for simple datasets. Where *Uniform* yields an accuracy of 0.90 after approximately 200 samples, the other techniques require at least 800 samples to reach the same performance. However, when the dataset complexity increases, it performs more similarly to the other techniques. The *Correct* labelling technique yields very low performances, i.e. an accuracy of less than 0.40, for simple datasets since predefined boundaries are simply confirmed.

Interestingly, when using a more complex dataset like MNIST, *Correct* outperforms the other techniques. It converges towards an accuracy of 0.85, whereas *Uniform*, *Modified* and *Sampled* all converge around 0.75. *Conditional Prior*, on the other hand, yields an accuracy of 0.15. Especially *Correct* yields a big increase in performance when the complexity of the dataset increases. The authors speculate that the low performance for simple datasets is caused by the fact that the learned boundaries are simply reinforced and no new classes are learned. However, when the feature space is very complex, the model might have not learned all of its features yet and is, therefore, able to adapt better to changes. While *Uniform* performs the best on simple datasets and relatively well on complex datasets, it yields a dip in performance before convergence for the CIFAR10 dataset, which can cause a problem in deployment.

In conclusion, *Uniform* performs best on simple datasets, while *Correct* performs best on complex datasets. Next to that, the *Uniform*, *Sampled* and *Modified* techniques all perform relatively well on complex datasets.

The study by Lucas et al. is relatively new, which also means that no other studies have compared these techniques. Additionally, Lucas et al. only use binary feedback, which yields less information when negative feedback is given.[11] Therefore, techniques that include negative feedback, i.e. all techniques except for *Correct*, might perform better in this study than in the study by Lucas et al.

3.2.2 Partial Label Learning

PLL is a sub-field of weak supervision in which a partial label is defined as "a set of candidate labels, among which only one is valid" [80, p. 47]. It is generally used in crowdsourcing, where many people label one sample. The goal of PLL is to appropriately deal with the different responses, i.e., candidate labels. In this study, an annotator is asked to give feedback on a number of classes. If they indicate that none of these classes is correct, it can be assumed that the correct label is among the remaining labels. Therefore, the problem in this study can be viewed to match PLL in the sense that the remaining labels are treated as candidate labels in PLL. However, it must be noted that only part of the PLL approaches are applicable in this study. PLL is generally used in situations where all candidate labels are (somewhat) similar.

[11] If a negative response is given for binary feedback, only one label is confirmed incorrect, meaning that all but one labels remain candidates. On the other hand, this study also tests requesting multiple classes, which means that more labels would be confirmed incorrect, and fewer candidate labels remain.

For example, if many people are asked to give a label to an image of a cheetah, answers such as *jaguar* and *leopard* may appear, but it is unlikely to yield the label *turtle*. Some PLL approaches use this information in their approach. This is, however, not applicable in this study, where the remaining labels of a feedback question are generally unrelated.

Additionally, most PLL approaches assume a maximum of three to four candidate labels, whereas all but one class could be candidate labels in this study. Thus, there are some differences between PLL and the LF in this study, but the idea of handling multiple candidate labels is the same, and therefore, some of the approaches are relevant for this study. While many PLL techniques have been proposed [81]–[90], only two techniques will be described. These two techniques perform significantly better than classical PLL approaches, e.g. [91]–[95], and are most relevant to this study. This is especially because they can be applied directly to a neural network, without requiring a model change, they are relatively simple to implement and perform well.

First, Seo and Huh propose DNPL (Deep Naive Partial Label learning) [96]. DNPL consists of a deep neural network but uses the naive assumption that all candidate labels have a uniform distribution because the ground truth is unknown. This is essentially the same approach as the *Uniform* approach by Lucas et al. [79] (described in Chapter 3.2.1), except that Seo and Huh explicitly define their loss function as logarithmic, whereas Lucas et al. have not defined their loss function. The advantage of this technique is that it is simple and their code is available online. They have not mentioned the number of candidate labels their approach was tested on, so it is unclear how well the technique will perform for larger partial labels.

Yan and Guo [97] propose PL-BLC (Partial Label learning with Batch Label Correction), which can be seen as an extension to the *Conditional Prior* technique from Lucas et al. [79] (described in Chapter 3.2.1). However, instead of simply using the prior probabilities as the label for each of the candidate labels, Yan and Guo add the prior probabilities to a variable, defined as 0 (in case of a non-candidate label) or 1 divided by the number of candidate labels (in case of a candidate label). Afterwards, the probabilities are normalized. This model is seen as the student model. Moreover, they implement a teacher model, of which the parameters are updated as the exponential moving average of the student model. The teacher model, which is less affected by the false candidate labels, is used to calculate the loss of the student model. They have shown that PL-BLC yields a high performance for 1 to 2 candidate labels, but have not tested the approach for more candidate labels. Additionally, Seo and Huh have shown that DNPL outperforms PL-BLC in some datasets and performs similarly in others [96].

3.2.3 Complementary Label Learning

Instead of focusing on the candidate labels, it is also possible to focus on complementary labels; labels which are confirmed to be incorrect. This is what CLL does; learning from confirmed false labels. In closed-set situations, i.e. no unknown classes appear after training, this is essentially the opposite of PLL.

Classical CLL only considers a single complementary label per sample. Only two studies [98], [99] have been found that tackle the problem of handling MCLs (Multiple Complementary Labels). First, Cebron et al. [98] apply SVDD (Support Vector Domain Description) which describes a set of objects and can recognize whether new objects fall inside or outside of this description. They model each class using an SVDD and base it on all negative label examples of that class. This is similar to the use of a one-class Support Vector Machine [100]. However, while a Support Vector Machine aims to maximize the distance from the samples from two classes to a hyperplane, an SVDD aims to find the minimum radius and location for a hypersphere such that all samples are captured in the space. After an optimal hypersphere has been found for each class, new samples are classified by finding the SVDD which has the biggest distance from the data point since the SVDD describes the negative label space of a class.

Next to that, Feng et al. [99] propose two techniques: using wrappers or an URE (Unbiased Risk Estimator). Wrappers simply decompose MCLs into multiple singular complementary labels. Feng et al. propose two different techniques for this. In the first approach they shuffle the training samples, after which in each batch, each sample is decomposed into multiple separate complementary labels. The second approach performs the same steps, but in a different order: they first decompose each sample into multiple separate complementary labels, after which they shuffle the training data. Both approaches yield similar performances. However, since classical CLL techniques are not developed for MCLs, information from the labels is diluted for both approaches. Using a URE aims to fix this problem by using MCLs as a whole, instead of splitting them up. Since the exact labels are not known, the loss function is replaced by a URE, which estimates the loss of a model. Feng et al. define their URE as follows:

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n}\Big(\sum_{y\notin\bar{Y}} L(f(x_i),y) - \frac{m-1-|\bar{Y}|}{|\bar{Y}|}\sum_{y'\in\bar{Y}} L(f(x_i),y')\Big) \tag{1}$$

where n is the number of samples in a batch, $\bar{Y}$ are the complimentary labels for sample x_i, L is the loss function, $f(x_i)$ the prediction of the model used and m the total number of labels. They compared multiple loss functions and found that the URE performs best with an upper-bounded loss function, i.e. logarithmic or exponential:

$$L_{EXP}(f(x_i),\bar{Y}) = exp\Big(-\sum_{y\notin\bar{Y}} p_\theta(y|x_i)\Big) \tag{2}$$

$$L_{LOG}(f(x_i),\bar{Y}) = -log\Big(\sum_{y\notin\bar{Y}} p_\theta(y|x_i)\Big) \tag{3}$$

While wrappers perform well for MCLs, using a URE with an upperbounded loss performs even better.

3.2.4 Conclusion

The main advantage of using a probability distribution is that it was made specifically for the task of handling LF (limited feedback). While Lucas et al. [79] only tested this approach

for binary feedback, they explored the same situation; yielding multiple candidate labels when receiving negative feedback. A disadvantage is that their approach was not compared to others, making it difficult to compare. PLL is generally focused on using relatively small partial labels, i.e. 2 to 3 candidate labels, and it is unknown how the performance of PLL changes for bigger partial labels. Similarly, CLL is not performed often for MCLs, but the two studies that have researched this [98], [99] have shown promise. Thus, it is unclear how well a probability distribution, PLL or CLL technique would perform in combination with LF. However, they are developed for handling multiple candidate labels suitably and are therefore relevant techniques to test.

Retrieving LF, as opposed to requiring hard labels, decreases the time and effort required from annotators. A different technique for decreasing the label task is through AL (Active Learning), as will be described in the next section.

3.3 Active Learning

3.3.1 Query Strategy Types

As explained in Section 2.4, AL is used to choose which sample should be queried first. Specifically, the query strategy determines the method of choosing samples. Therefore, this section will discuss query strategies from related work.

First, it is important to note that two studies [101], [102] have proposed solutions for AL with soft labels specifically, although their approaches use assumptions which are not applicable to this study. Li et al. [101] propose a solution to AL with partial labels. However, they assume that the system always returns a soft label, as opposed to having the chance of retrieving a hard label. Next to that, they use a disambiguation strategy, i.e. the real label is disambiguated from the partial label, which allows them to use standard AL strategies since they assume that every query results in a hard label. This is different from the methods applied in this study, as can be seen in Section 3.2. Moreover, Hu et al. [102] propose a solution to AL with binary feedback. In their study, all labels are structured hierarchically, where binary questions can be asked to yield more information about the label. The question with the greatest expected reduction in entropy is selected. This does require a hierarchical structure of labels, which is not applicable in this study. Since these methods are not applicable in this study, regular AL query strategies will be discussed in this section. First, an overview is given of the types of query strategies.

There are three types of AL query strategies: based on heterogeneity, performance, representativeness and hybrid methods. Heterogeneity-based query strategies attempt to select samples which are heterogeneous to the existing labelled samples [103]. The most common methods are uncertainty sampling, query-by-committee and measuring expected model change [49].

A downside of heterogeneity-based sampling is that it can lead to the selection of outliers because outliers are generally the most uncertain samples. Therefore, performance-based

query methods attempt to improve the performance of a model on future data points. Thus, where heterogeneous-based methods focus on the expected performance of one sample, performance-based methods focus on future performance by measuring the performance on a validation set [103]. Two examples of performance-based query methods are estimated variance reduction and error reduction [49].

Representativeness-based query methods attempt to find samples which model the underlying space of the training set [103]. Therefore, representative sampling tries to compress the training set into a smaller set of points which can be queried. This problem is described as the core-set problem, where the goal is to find a subset of the training set, which will function as a proxy for the full training set [52]. Lastly, Hybrid query strategies combine two or three of the previously named methods, e.g. the Cluster-Margin approach [52].

Due to time constraints, it is infeasible to compare many AL methods in this study. Therefore only one field will be discussed in this section. The choice was made to focus on heterogeneity-based methods because they generally perform relatively well for their simplicity and are often used as a comparison in state-of-the-art methods. Therefore, this section will focus on three heterogeneity-based methods, namely uncertainty sampling, query-by-committee and expected model change.

3.3.2 Uncertainty Sampling

One of the simplest and most common query strategies within AL is uncertainty sampling. This technique selects the samples which it can predict with the least certainty. The main advantage of uncertainty sampling methods is that they are easy to implement and fast [104]. While there are many ways to measure uncertainty, e.g. margin [50], [52], [104]–[106], least confidence [49], [52], [104], [105], best-versus-second-best [107] and Gini-index [103], the most common method is entropy [49], [52], [104]–[106], [108]. Entropy measures the uncertainty over all possible classes and is defined as:

$$H(x,\theta) = -\sum_{i}^{m} p(y_i|x;\theta)\log p(y_i|x;\theta) \tag{4}$$

where x is a sample, m the number of classes and $p(y_i|x;\theta)$ the estimated probability of assigning class y_i for sample x given model parameters θ.

The sample with the highest entropy is selected to be queried. This results in the samples with the most spread probability distribution being chosen. Gal et al. [109] use entropy within a deep learning model, where $f_i(x)$ is defined as the softmax output of the model. While the softmax output does not consist of probabilities in itself, they can be used as such, because they do add up to 1. There are different methods to calculate $f_i(x)$, for example by taking the proportion of neighbours votes as a probability measure [50]. However, softmax is the most straightforward method when using neural networks.

3.3.3 Query-By-Committee

The QBC (Query-By-Committee) approach is another well-known approach based on measuring uncertainty and uses a committee of classifiers [49], [103], [104], [106]. Each classifier is trained on the same training set but has different hypotheses. For example, Figure 6 shows how multiple models can return different decision boundaries. After constructing a committee, each classifier votes on its decision. The sample which yields the most disagreement will be queried. Seung et al. [110] first proposed QBC, where they calculated the entropy over the number of votes for a class. However, also measures such as the gini-index, KL-divergence, Korner-Wrobel disagreement and Jensen-Shannon divergence have been used as QBC measure [103], [108]. The advantage of using QBC is that it is easy to understand [104], but it does require multiple classifiers, which increases the complexity of the system and makes it more computationally expensive.

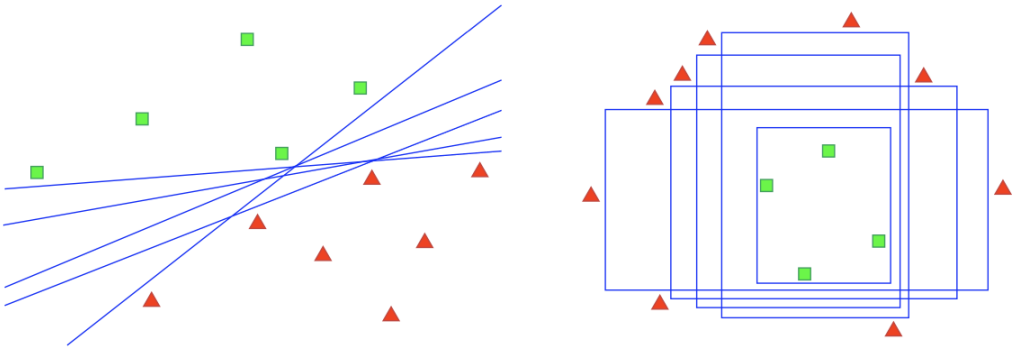


Figure 6: Two sample spaces with possible decision boundaries from multiple models. Source: [49]

3.3.4 Expected Model Change

The EMC (Expected Model Change) approach bases its queries on the samples that would change the model most if they were added to the training set. Settles et al. [111] propose calculating the EMC by EGL (Expected Gradient Length), as follows:

$$EGL(x_i, \theta) = \sum_{i}^{m} p(y_i|x_i;\theta)||\nabla L(T^{+(x_i,y_i)};\theta)|| \tag{5}$$

where x_i is a sample, m the number of classes, $p(y_i|x_i;\theta)$ the estimated probability of assigning class y_i for sample x_i given model parameters θ, the loss function L, and $T^{+(x_i,y_i)}$, the labelled training set T with sample (x_i, y_i) added to the set.

The advantage of EGL is that it selects the samples which will influence the model most, regardless of the label [49]. However, it is quite computationally expensive [104]. It is also important to note that this method is only applicable to models using gradient-based training and weights.

3.3.5 Conclusion

Three AL query techniques were discussed in this section: uncertainty sampling through entropy, QBC and EMC. The main advantage of uncertainty sampling through entropy is that it is relatively easy to apply, while it is still often used as a comparison in state-of-the-art methods. On the other hand, QBC and EMC are both more computationally expensive. Therefore, uncertainty sampling through entropy is a recommended method to use in this study.

3.4 Conclusion

This chapter has described DCS and ND methods, how to incorporate LF and how to apply AL. Two DCS approaches have shown promise to use in this study: k-NN with OLA/LCA and decision space with OLA/LCA. Additionally, recommended methods for ND in this study are a k-NN with a threshold, a k-NN with a comparison of the distance to the neighbours' neighbours and uCBLOF.

Afterwards, the concept of LF has been explained, i.e. that not every sample yields a hard label, but instead, some yield soft labels. Three relevant fields have been discussed, which are all promising to use, although none has been used in this specific situation: probability distribution, PLL and CLL. Lastly, AL strategies were discussed, which can further limit the labelling efforts. This section has discussed three heterogeneity-based techniques, from which uncertainty sampling through entropy seems like the most suitable method to use.

The next chapter will use the DCS and ND methods to define a framework, which will be used during the experiments. Afterwards, the knowledge on LF and AL will be used in Chapter 5, which will focus on the methodology of this study.

4 Framework Design

As described in Chapter 1, personalization will be achieved through a mixed model. While most other literature on personalization uses other methods, a mixed model is less likely to overfit when few samples are available and is more transparent in the sense that it can show which model was responsible for a given prediction. This mixed model requires two components, as described in Section 2.2 and 2.3: ND (Novelty Detection) and DCS (Dynamic Classifier Selection). DCS allows the personal model to become an expert in areas of the input space where the general model lacks performance rather than having to learn about the full input space. Additionally, ND is required to find whether the DCS component has enough knowledge of the dataset to make an informed choice. To ensure that all components can interact well with each other, a framework was set up, as shown in Figure 7. Please note that Figure 7 is a simplified version of the framework and Figure 8 shows the full framework in a formal notation. This section will describe the integration of each component in the framework. First, the need for each component will be explained, after which the components and their interactions will be explained formally.

4.1 Integration of Components

The framework consists of two phases: a prediction and a training phase. In this study, samples arrive continuously, and thus, predictions are made continuously. Training only occurs periodically, when samples are selected through AL after a certain time period. For example, if training occurs daily, the most useful samples are selected from all samples collected during the past day.

4.1.1 Prediction

The mixed model consists of a general model and a personal model, of which one is chosen to predict the sample. First, ND is used to detect whether the incoming sample is similar to the previously seen samples, and thus, if the DCS component can make a reliable estimation of the performance of the models for the sample. If ND classifies an incoming sample as novel it is assumed that the general model is the safest option since the personal model might not have been trained on similar data. On the other hand, if the incoming sample is not novel, DCS is used to select the best-fitting classifier.

4.1.2 Training

The training phase occurs periodically and consists of an AL component which chooses the best sample to label out of all samples encountered in the past time period.[12]

[12] All samples before that time period, excluding the samples on which feedback was gathered, are discarded. This choice was made because of two reasons. First, in most situations, it is infeasible to save all incoming data. Therefore, data generally has to be deleted after a certain amount of time. Second, it is beneficial that an annotator remembers the occurrence of the sample to increase the chances of receiving the correct label.

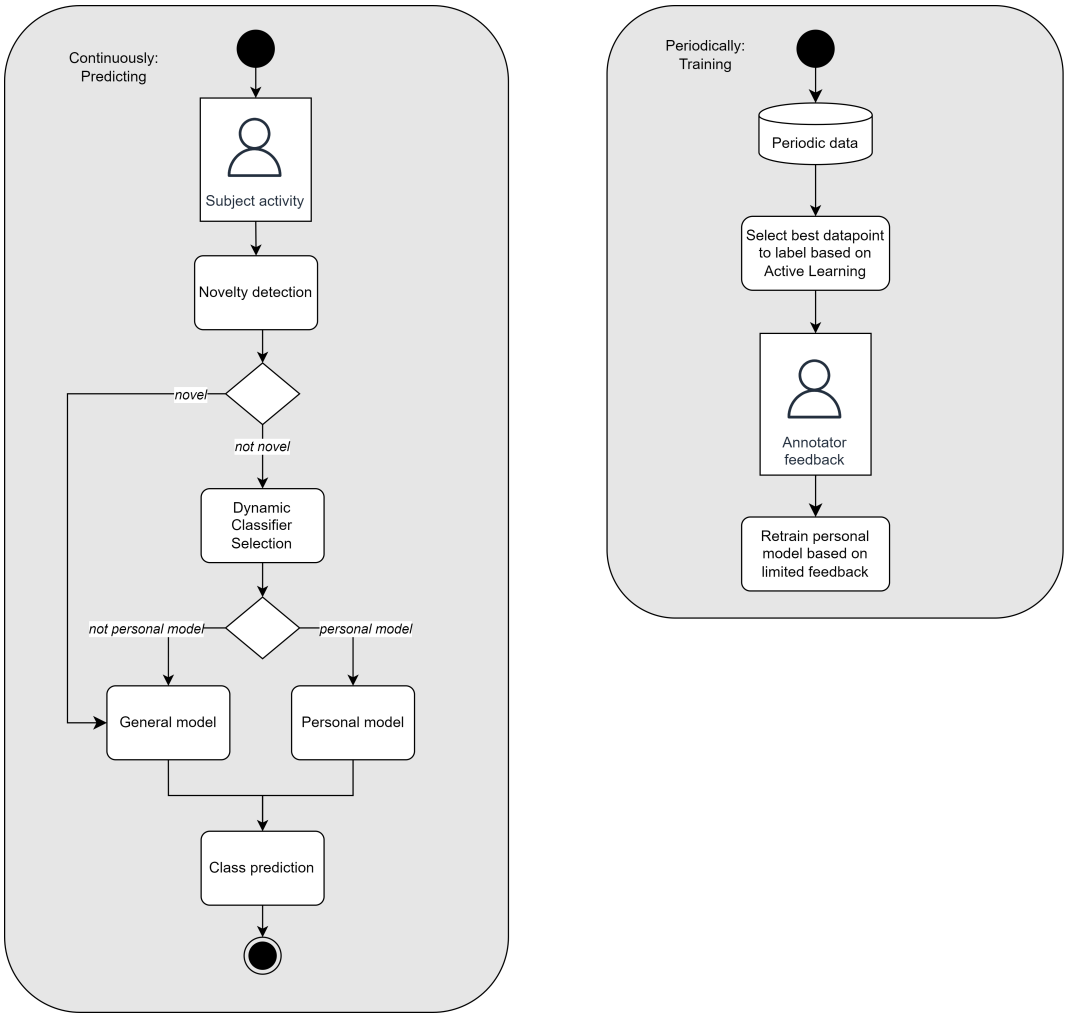


Figure 7: A simplified version of the proposed framework.

The goal of the AL component is to select the most informative sample from the samples gathered during a certain time period. This sample is then sent to the annotator for feedback, along with the classes with the highest output scores. The annotator is then asked to select the correct label, or indicate that the correct label is not among the requested labels. In case the annotator indicates that the correct label is not among the requested labels, a soft label is returned, therefore the actual label of the sample is not known. Instead, it is only known that the real label is among the list of labels that were not presented to the annotator.

All feedback samples are randomly split into two datasets, where 33% of the data is used as a comparison set for ND and DCS and 66% as a train set.[13] A comparison set was included because it is not recommended to use the training set for both training the personal model and evaluating the personal model during DCS. If the personal model trains on the

[13]The 33% and 66% split was based on practical considerations. Increasing the portion of the comparison set would lead to a larger amount of feedback samples to train the model. On the other hand, decreasing the proportion of the comparison set would result in having insufficient data to perform DCS.

same data it will be evaluated upon, it could lead to an unfair preference for the personal model during DCS due to data leakage. Instead of using a comparison set, it would also be possible to use the training set for DCS but leave out some of the neighbours and retrain the model until all neighbours have been evaluated. However, this leads to infeasible training times. Therefore, the choice was made to split the feedback samples into a training set and a comparison set. A disadvantage of this is that it requires more feedback samples to train the model.

4.2 Formal Definition of Components

After explaining the process of prediction and training, the formal definitions of all components and their interactions will be given. Both the prediction and training phases are denoted with the following formal notations in Figure 8. Let (x_i, y) be an incoming sample, where x_i is sampled from $X \in \mathbb{R}^v$, which denotes the feature space with v dimensions.

The label of x_i is defined as $y = \{y_i | i = 1, 2, ..., m\}$, where m denotes the number of classes, $b \in \{1, 2, ..., m\}$ denotes the index of the label, and

$$y_i = \begin{cases} 1, & \text{if } i = b \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

The data used to train and predict is saved in feedback set F, consisting of training set T and comparison set C such that $F = T \cup C$. The generation of the sets and representation of the samples of these sets will be described in the next sections.

4.2.1 Prediction

The goal of the prediction phase in the framework is to predict the class of x_i correctly so that the prediction matches y. To predict the label of x_i, a personal model and a general model are combined through a mixed model. The general model can be seen as a function $g(x_i)$, which outputs o, consisting of a softmax score for each class. The predicted label for the model is given as $\hat{y} = argmax(o)$. Similarly, the personal model can be seen as a function $p(x_i)$, which yields the same type of output.

The goal of DCS is to find c, which is a boolean output, indicating whether it is beneficial to select the personal model for predicting x_i or not. The DCS component bases its choice c on the performance of the classification functions p and g on similar samples to x_i in the comparison set C. The procedure of selecting and evaluating similar samples will be explained in Section 4.4.

ND is used to detect whether x_i is similar to the samples in the comparison set C to indicate whether the DCS component has enough information to make a reliable estimation of the performance of the models for x_i. The ND component requires knowledge of x_i and C and outputs a boolean result n, i.e. *novel* or *not novel*. In case n is *novel*, the choice in classifier c is set to *not personal model*, while if n is *not novel*, DCS determines the output of c. Section 4.3 will elaborate on the ND method used in this study.

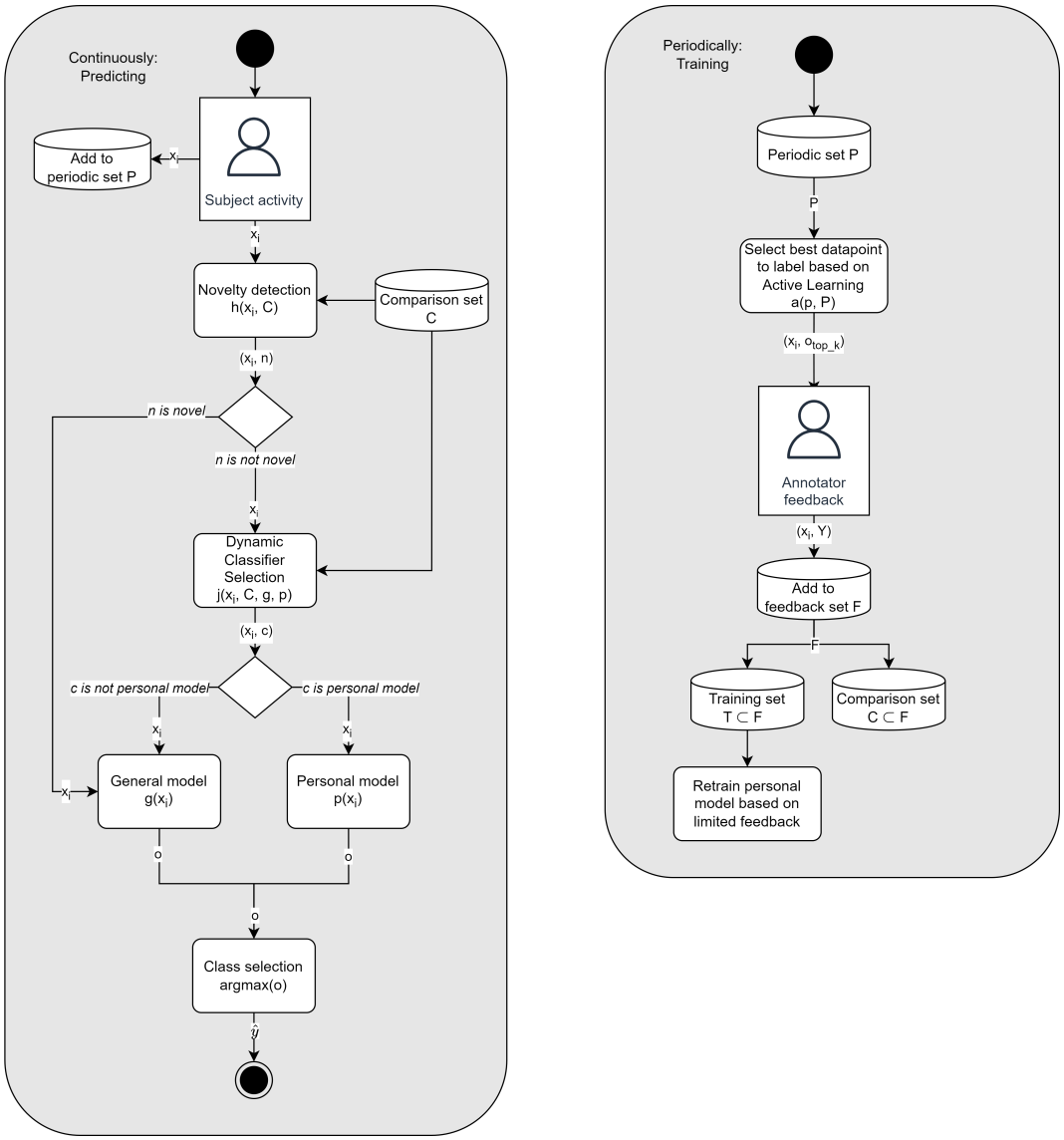


Figure 8: The proposed framework.

4.2.2 Training

The training process occurs periodically and uses periodic set P, which consists of $[x_{t-s}, ..., x_t]$, where s is a certain time period and t is the current time frame. The most informative sample, i.e. x_i, is chosen out of P through AL function a. AL function a bases its choice on the samples in periodic set P and the personal model p. The specific metric to calculate informativeness used in this study is explained in Section 5.1.3.

The selected sample x_i is sent to the annotator for feedback, along with the e classes with the highest output scores. The value of e varies during the experiments, where $e \in \{1,...,m\}$.[14] The annotator provides LF (limited feedback) on this sample so the actual label y is not always known. Instead, a soft label may be returned, which yields $Y = \{Y_j | j = 1,2,...,m\}$, where m denotes the number of classes, $I \subset \{1,2,...,m\}$ denotes the indices of the candidate labels, and

$$Y_j = \begin{cases} 1, & \text{if } j \in I \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Additionally, in case $|I| = 1$, a hard label is received and $Y = y$. In other cases, i.e. $|I| > 1$, a soft label is received and it is only known that $\exists j : (y_j = 1) \wedge (Y_j = 1)$, thus that the real label is among the partial labels. An LF technique is used to handle labels where $|I| > 1$, as will be explained in Section 5.1.2.

After receiving feedback, (x_i, Y) is stored in feedback set F. Afterwards, feedback set F is split up into comparison set C, used for DCS and ND, and training set T, used for training the personal model. The model is then retrained based on T. The following sections will describe the implementation of the aforementioned components, starting with ND.

4.3 Novelty Detection

ND can be accomplished through various methods, as described in Chapter 3.1.2. In the framework, the most important requirements for the novelty detector are:

1. It should **not require the label of the test sample**. The method should be able to detect outliers solely on the data structure of the test sample since the label is not known (yet).

2. It should handle **high-dimensional data** relatively efficiently. As described in Chapter 1, HAR (Human Activity Recognition) data is often high-dimensional, so the method must be able to handle high-dimensional data in a reasonable time period.[15]

3. It should perform well for **a small number of samples**. As described in Chapter 1, the method should work well in the early stages of deployment, even when few feedback samples have been gathered.

4. It should perform **globally**. As described in Section 2.3, it is assumed that the classifier can still make reasonable predictions on locally novel samples, so only globally novel samples are considered.

5. As described in Section 2.3, this study aims to find individual samples which are dissimilar to the others, thus it should focus on **point novelty** samples.

[14] Section 5.1.2 provides more information on the variation in e during the experiments.
[15] Please note that it is not required that the method runs in real-time for this study.

The related work in Section 3.1.2 has shown that a k-NN (k-Nearest Neighbours) is the recommended method as ND component.[16] Additionally, it also complies with the requirements set.

The k-NN is used to calculate $d_k(x_i)$, which is the average distance to the k-nearest neighbours of x_i. As will be explained in-depth in Chapter 5, two datasets will be used in this study: a toy problem and a HAR dataset. The value of k depends on the dataset used and has been found through exploratory tests for the toy problem and HAR dataset separately. For the toy problem, a new dataset was generated on which the tests were performed, where k=15 seemed most suitable. For the HAR dataset, the validation data for training the model was used to find k=2. Importantly, none of these tests involved using the test set. The cosine was used as the distance function because it is often used in studies with high-dimensional data [62] and performs similarly to the Euclidean distance [112].

Similarly to [70], $d_k(x_i)$ is compared to a set threshold and all samples above this threshold are classified as novel samples. They based this threshold on the top 2% of all distances, such that 2% of the data is considered a novel sample. However, in this study, it is not reasonable to assume that the percentage of novel samples is always the same. Leys et al. [113] advise to use the deviation around the median as a measure for detecting novel samples.

This uses the MAD (Median Absolute Deviation), which is defined as follows:

$$MAD = w * median(|d_k(x_i) - median(d_k(x_C))|) \tag{8}$$

where $d_k(x_i)$ is the average distance from sample x_i to its k-nearest neighbours, $d_k(x_C)$ contains the average distance from each sample in comparison set C to their k-nearest neighbours and w is a constant, usually set to $w = 1.4826$ [113]. Please note that $d_k(x_i)$ is a single value and $d_k(x_C)$ contains $|C|$ values. The decision criterion is defined as follows:

$$n = \begin{cases} novel, & \text{if } d_k(x_i) > median(d_k(x_C)) + q * MAD \\ non-novel, & \text{otherwise} \end{cases} \tag{9}$$

where q defines the conservativeness of classifying samples as novel. In this study, it is preferred to classify samples as novel quite conservatively, since this automatically assumes the general model is better. Therefore, q is set at 3, which is defined as *very conservative* [113].

Additionally, if $|C| < k$, k is set at $k = |C|$, so that ND and DCS can still be used if the number of samples in C is small. Furthermore, as the comparison set grows in size, it is preferred not to unnecessarily classify samples as novel. To prevent this, it is assumed that none of the samples is novel after at least k samples have been gathered for each class.

[16] While it would be interesting to study, it is out of scope for this study to compare multiple methods in terms of their performance.

4.4 Dynamic Classifier Selection

If the novelty detector has established that a sample is not novel, DCS is used to choose a classifier based on the estimated performance of both models on the sample. The DCS component has the following requirements:

- It should output **one of the classifiers** and not weight the predictions of both classifiers. This makes it more transparent in terms of which classifier is responsible for the prediction.
- Similarly to the novelty detector, it should be able to handle **high-dimensional data** relatively efficiently.
- Similarly to the novelty detector, the DCS component should perform well for **a small number of samples**.

As explained in Section 3.1, the k-NN with LCA (Local Class Average) is recommended for DCS in this study, which also complies with the requirements set. Figure 9 shows the steps taken to perform DCS in this study. The input of the DCS component consists of the test sample x_i and the comparison set C. The output of the component is a binary indicator of the classifier used.

As mentioned previously, the k-NN uses the same settings for ND and DCS, i.e. $k = 15$ for the toy problem and $k = 2$ for the HAR dataset and cosine as distance function. Both the general and personal models are then tasked with predicting the labels of the k-nearest neighbours, which is used to calculate the LCA. The classifier with the highest LCA on the k-nearest neighbours is chosen to classify x_i. Now that the framework design has been presented, the methodology of the experiments can be explained, which uses the framework as a baseline.

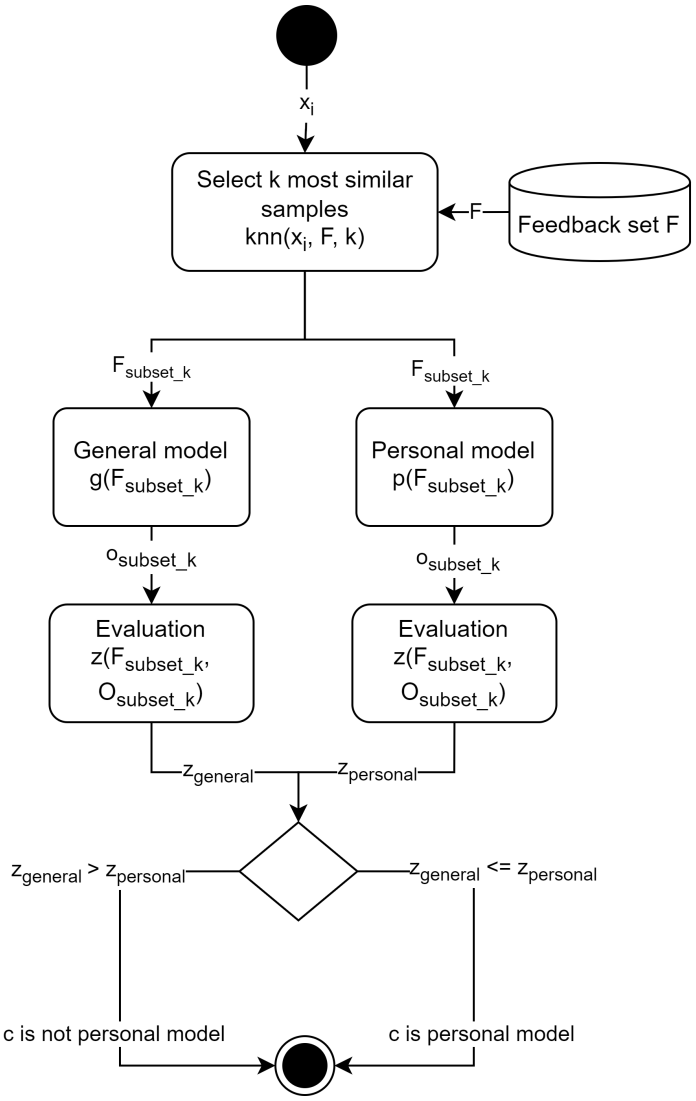


Figure 9: Selecting the best classifier for a sample x_i based on its performance on the k most similar samples.

5 Methodology

As explained in Chapter 1, this study aims to find an effective way of using LF (limited feedback). First, the trade-off between the maximization of information gain and the minimization of the level of detail of feedback will be researched, which will be determined by the use of LF. Second, the minimal number of feedback samples required to achieve an optimal level of performance by the effective use of LF will be researched, which will be tested by the use of AL (Active Learning). Therefore, two experiments will be performed: a comparison between five LF techniques and a comparison between two AL techniques in terms of their performance and adaptability. The methodology for these experiments will be described in this chapter.

To validate the results on two different datasets, the experiments will be performed on a toy problem and a HAR (Human Activity Recognition) dataset. Each of these datasets will be described individually, along with the required preprocessing steps and the hyperparameters for the personal and general models for each dataset. Lastly, a description will be given of the evaluation procedure and evaluation metrics.

5.1 Evaluation Components

5.1.1 Framework Evaluation

The components in the framework will be evaluated by measuring the accuracy of the general model and the personal model without and with the framework. Additionally, visualizations will be used to explain how the framework operates and why components are needed. The framework will be evaluated for a varying number of feedback samples, as will be further explained in Section 5.2.1 and 5.2.2.

5.1.2 Limited Feedback

In this study, feedback is requested by asking whether the correct label is presented in a number of classes. The number of classes presented to the annotator is referred to by e. If the correct label is within these classes, the annotator can simply select the correct label and a hard label emerges. However, if the correct label is not within this list of classes, the annotator can only signal that the correct label is missing. Therefore, the only information known is that the correct label is among a list of remaining labels, but not which label exactly. Such a label is referred to as a soft label, and typical ML (Machine Learning) algorithms are not equipped to handle these labels.[17]

The fields of PLL (Partial Label Learning), CLL (Complementary Label Learning) and using probability distributions can handle soft labels and have shown promise in Section 3.2.

[17] Please note that this process also occurs during DCS (Dynamic Classifier Selection), thus only a soft label might be present. In that case, the prediction of the model is regarded as correctly classified if the correct label is among the candidate labels.

From these fields the following LF techniques will be compared: the PLL technique *DNPL (Deep Naive Partial Label learning)* [96][18], the CLL technique *URE (Unbiased Risk Estimator)* [99] and the probability distribution techniques *Correct*, *Sampled* and *Modified* [79]. *DNPL* was chosen out of the PLL techniques because it performs well while being relatively easy to implement. From the three identified CLL techniques, i.e. *URE*, *wrappers* and *SVDD (Support Vector Domain Description)*, *URE* was chosen because it performs better than using wrappers but does not require a model change, which is required for *SVDD*. Lastly, *Correct*, *Sampled* and *Modified* all perform well, although their performance differs when the complexity of the dataset changes. The implementation of all techniques was unchanged from their original papers. The following paragraph will shortly explain how each technique works.

First, the *Correct* technique simply filters the training data such that all incorrectly classified samples are discarded as follows:

$$T_{Correct} = \{(x_i, \hat{y}) | \hat{y} = y, \text{ for } i = 1, ..., |T|\} \tag{10}$$

where $\hat{y}$ is the predicted label, y the real label and T the full training set.

The *Sampled* technique adapts the soft label Y for sample x_i and each class j as follows:

$$Y_j = \begin{cases} 1, & \text{if } j = q \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where $q \in I$ and $I = \{j | Y_j = 1\}$ is the set of candidate labels for x_i. Therefore, q is a random index of all candidate labels in Y. Through this, the *Sampled* technique randomly samples one of the candidate labels to be the hard label.

The *Modified* technique adapts the soft label Y for x_i and class label j as follows:

$$Y_j = \begin{cases} o_j, & \text{if } j \in \{I\} \\ -1000, & \text{otherwise} \end{cases} \tag{12}$$

where o_j denotes the softmax output of the model for x_i and class j and $I = \{j | Y_j = 1\}$ is the set of candidate labels for x_i. After the non-candidate class labels are set to a large negative number, which Lucas et al. [79] chose as -1000, all labels are normalized again with a softmax function. Further details of the implementation of these three methods can be found in [79].

Instead of adapting the label, the *URE* adapts the loss function, while the label Y is kept the same as in equation 7. The *URE* combines the loss of the candidate and non-candidate functions as follows:

$$\hat{R}(f) = \frac{m-1}{|\bar{I}|} L_{log}(f(x_i), I) \tag{13}$$

where I is the set of candidate labels for x_i, m denotes the total number of classes and L defines the underlying loss function, as defined in equation 14. In other words, the *URE* weights the loss of the model with the number of candidate labels.

[18] Please note that Lucas et al. [79] proposed the same technique under the name *Uniform* in their paper on probability distribution techniques.

The log loss performed best in their study, and is therefore also used in this study:

$$L_{log}(f(x_i), I) = -log(\sum_{y \in I} f(y|x_i)) \tag{14}$$

where I is the set of candidate labels for x_i. The implementation of the *URE* is equal to the implementation by Feng et al., so further details can be found in their paper [99].

The PLL technique, *DNPL* [96], is equal to the *Uniform* technique from Lucas et al. [79] and was implemented by setting the label as follows:

$$Y_j = \begin{cases} \frac{1}{|I|}, & \text{if } j \in \{I\} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where Y is the partial label for x_i, and I is the set of candidate labels for x_i. The loss function was set to the log loss, as depicted in equation 14. The differences between *DNPL* and *URE* are the form of the label and the weighting factor.

Importantly, a *Full-Feedback* model is used as a reference, and receives hard labels for every sample, as opposed to the other techniques. This *Full-Feedback* model is the same as the personal model used as a comparison for the framework, which also receives hard labels for every sample.[19]

In the experiments, the labels of feedback set F were based on the predictions of the previously trained model. For instance, the predictions of a model trained with 60 samples were used to make the labels to train a model with 80 samples. Additionally, to compare the methods, the number of requested labels e were varied to see how well each method performs on different partial label sizes.

5.1.3 Active Learning Technique

To test how the effective use of feedback determines the number of samples needed to yield optimal performance, an AL method was compared to random sampling. Section 3.3 has shown that using entropy is a feasible method for selecting samples effectively. This means that specific samples are queried from an annotator instead of sampling randomly from a distribution. The advantages of entropy-based sampling are that it is fast, easy to implement and yields a satisfactory performance in multiple studies [52], [104], [109]. Even though it might not be the best-performing state-of-the-art method, it is still often used in recent studies, where the difference in performance is generally minimal, i.e. at most a few percent difference in accuracy [52], [114], [115]. Since this study aims to show the effect of an AL technique, rather than compare the latest state-of-the-art techniques, entropy-based sampling is suitable for this study.

The framework, as shown in Figure 8, includes choosing samples from the periodic set P. In a real-time application, P would include the most recent time period. However, the

[19] Since all LF techniques could be used as a personal model, the *Full-Feedback* model was renamed to indicate a specific type of personal model which receives hard labels for every sample.

datasets used in this study, which will be explained in Section 5.2, do not contain information about the time period of the sample. Therefore, the samples were randomly split up into smaller pools of samples to simulate different time periods. After sampling a sample from the pool, the model is retrained and after the desired number of samples is reached, the samples in the pool are discarded and the procedure repeats with a new pool.

During training, the number of training samples was increased incrementally.[20] First, a pool of samples was used as a basis, consisting of 4 and 20 samples for the toy problem and HAR datasets respectively. This can help the entropy-based strategies in deciding which samples are the most uncertain in the personalized context. Afterwards, a small number of samples is collected from each pool using either random or entropy-based sampling. This small number of samples consists of one sample for the toy problem and ten samples for HAR.[21] To minimize the chance of entropy-based sampling selecting 10 similar samples, the strategies selected one sample at a time and were retrained after each selected sample.

Due to time constraints, the sampling techniques were only tested for one LF technique. To ensure that the performance of the AL techniques is not negatively affected by this technique, a well-performing LF technique is selected based on the LF experiments for the toy problem and HAR dataset separately.

5.2 Datasets

5.2.1 Toy Problem

A toy problem was created to increase the understanding of how the framework operates and perform experiments on a low-dimensional dataset with a small model. The dataset was created synthetically, because this makes it adaptable to specific needs, e.g. the level of overlap between classes. First, the dataset will be explained in-depth, after which the required preprocessing steps and model architectures for the personal and general models will be discussed.

[20] See Section 5.2.1 and 5.2.2 for more information on the training process.

[21] This corresponds with a 50% selection rate for HAR since each pool of samples consists of 20 samples. A 50% selection rate is relatively high, which is generally an undesirable property in AL. A high selection rate increases the chances of randomly picking the most informative samples, which decreases the difference in effectiveness between the AL strategies. However, decreasing the selection rate was, most likely, not a possible solution. Training a model on too few samples could lead to a decreased performance, which, in turn, leads to more soft labels for the next round of training. This could result in a continuous decline in performance. Therefore, the choice was made to use a relatively high selection rate to prevent this.

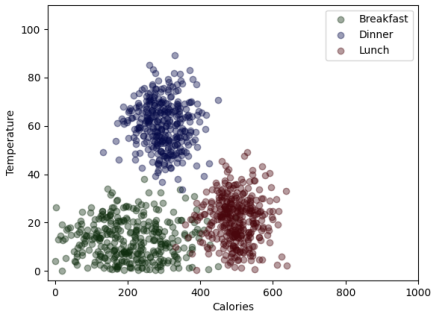
Data Generation The toy problem dataset is aimed to match the characteristics of a personalized HAR dataset, while still being simple enough to illustrate the workings of the framework. Therefore, it has the following requirements:

- A general dataset and an adapted **personalized** version.
- A **nominal classification** goal.
- Low-dimensional and relatively easy to classify samples to keep the problem **simple**. This allows the focus to be on the personalizability of the model and handling feedback, rather than the classification itself.
- A **balanced** dataset, such that the classes are distributed evenly. This choice was made because, as described in Chapter 1, this study does not focus on the imbalance of classes that often occurs in HAR datasets.
- **No noisily** labelled samples. This dataset assumes that all samples are labelled correctly, as described in Chapter 1.

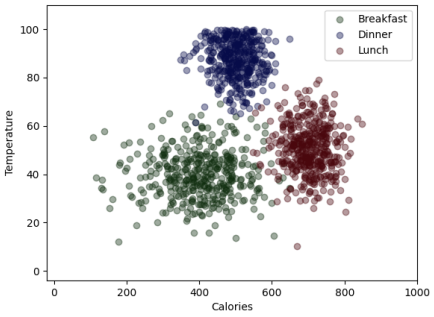
Modelling a HAR dataset for this toy problem, which generally contains quite complicated features and many features, would over-complicate the dataset and would make it relatively difficult to visualize it. Therefore, the choice was made to model meal preferences, since these can be summed up in a few, easy features, but also contain personal differences. For example, for the general population, the meal preferences of *breakfast*, *lunch* and *dinner* can already be differentiated by the temperature and number of calories in the meal. This simplified problem allowed the focus to be on the personalizability of the model and handling feedback, rather than the classification itself. Therefore, the dataset consists of two features, i.e. temperature and the number of calories, and three nominal output classes which specify the type of food, i.e. *breakfast*, *lunch* and *dinner*.

The personalized dataset is an adapted version of the general dataset so that some personal samples are labelled differently than the general samples. The goal of the framework is to adjust quickly to these changes in the personal dataset. To fully demonstrate the workings of all components in the model, and to validate its use, multiple versions of the dataset were created. This was done to show the robustness of the method and to create situations in which components excel or lack. These three adaptions are drift, unexplored and combined and are visualized in Figure 10. Each of these adaptions was motivated by problems that can be seen in HAR, as will be explained for each adaption separately.

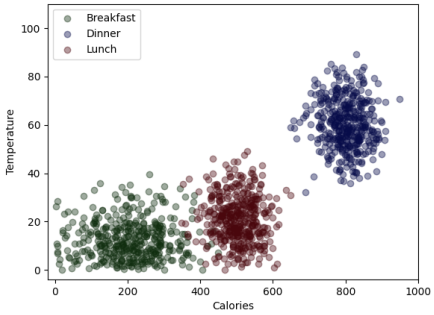
First, it is possible that some (or all) classes in a personal dataset contain samples with drift in comparison to the general data, as can be seen in Figure 10b. This can be seen in HAR, for example, when a person's well-being is decreasing due to a disease such as Alzheimer's [116]. While this is within-subject drift, this indicates that between-subject drift is also a likely possibility.



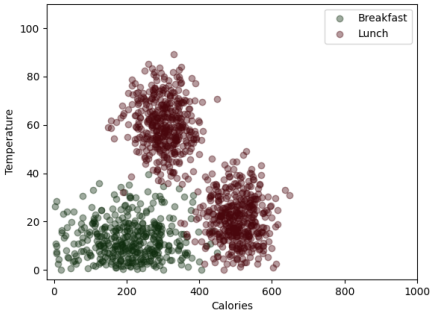
(a) General dataset.



(b) Drift: personal distribution contains drift in comparison to the general dataset.



(c) Unexplored: personal distribution in which the input of one class is shifted towards a previously unexplored area.



(d) Combined: personal distribution in which the outputs of two classes are merged.

Figure 10: The general dataset and three personalized datasets.

Second, it could also occur that data points from one of the classes shift towards a previously unexplored area of the input space, as can be seen in Figure 10c. Research has shown that there is a high variation in how people perform activities [20]. This may lead to a person performing an activity in a way that has not been encountered before.

Third, it could also occur that data from a certain input is now mapped to a different output, as can be seen in Figure 10d. This is motivated by the fact that when general a model is trained on healthy people, it can lead to a decrease in performance when used for people with a disability or illness because they perform the activities differently than expected [117]. While it would be possible to train a general model on people with disabilities and illnesses as well, it would be difficult to include all variations in the population. For example, for a person with just one leg, walking with a walking stick and jumping on their leg might both be considered walking. However, for the general population, one would be considered walking and another jumping. While these adaptions may not occur for every new subject, they can help to show the promise and flaws of the model architectures.

The datasets were created using a normal distribution with a different mean and standard deviation for each class, for which the values can be found in Appendix A. Moreover, each

variable was truncated, so that the minimum temperature and minimum number of calories is 0 and the maximum temperature is 100. The classes overlap slightly, but not too much, to avoid over-complicating or oversimplifying the problem.

The size of feedback set F ranged between 4 and 100 samples to get better insights into the training process.[22] Feedback set F was then split up randomly into training set T, with 66% of the data, and comparison set C, with 33% of the data. To get better insights into the variation in performance, and due to the relatively small dataset size, five-fold cross-validation is used. Additionally, the data for both the general and personal datasets was normalized using min-max normalization, as can be seen in equation 16, before it was used as input into the models.

$$x_{norm\_i} = \frac{x_i - min(F_x)}{max(F_x) - min(F_x)} \tag{16}$$

where F_x contains all x-values in feedback set F, and x_i is the sample which is normalized.

General and Personal Model The mixed model used in this study consists of a general and a personal model, which are both the same in terms of structure. For the toy problem, these are both neural networks, which were kept small to prevent overfitting. The neural networks consist of 2 densely connected layers, namely an input layer with 2 neurons and an output layer of 3 neurons, without any hidden layers. The output layer uses a softmax activation function. Adam was used as an optimizer with a learning rate of 0.01, a categorical cross-entropy loss and a batch size of 4. After training the general model, the personal model inherited its weights.

33% of training set T is used as validation data. Since a batch size of 4 is used, the validation loss is only calculated after gathering at least 4 validation samples, i.e. after gathering at least 12 training samples. The training is run for 50 epochs, after which the best model is selected based on validation loss. This method was chosen, as opposed to early stopping, because it is a more reliable method if the loss curve is unsteady. While choosing the best model after all epochs requires an increased training time, it does ensure that the best model is chosen. Tensorflow 2 [118] was used to build the models.

5.2.2 Human Activity Recognition

All methods are also tested on a HAR dataset, namely the NTU RGB+D 120 dataset [29]. This dataset is not focused on personalization, so it was adapted to include personalization in it. This chapter will first describe the NTU dataset, after which the steps taken to personalize the data are discussed. Lastly, the architectures of the personal and general models are discussed.

[22] Specifically, the models were trained with 4, 6, 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 samples.

Dataset Selection and Personalization The HAR dataset was selected using the following requirements:

- Actions should be performed **a reasonable number of times per person**. Since it is important to have enough data on a subject to personalize and test the model, the dataset should contain a sufficient number of samples per subject.
- As described in Chapter 1, this study focuses on balanced datasets. Therefore, the activities in the HAR dataset should be **balanced**.
- The activities should only include **individual actions** and no interactions between multiple persons. While interactions are also interesting to study, it is out of scope to test actions including interactions in this study.
- The dataset should include **3D joints of a skeleton** to match the data type of the mmWave activity recognition pipeline.
- Collecting skeleton data is performed with a sensor, which can have faulty readings. The dataset should contain **high-quality joint estimations** and **few faulty readings** so that the results are not highly influenced by inaccuracies in the dataset.
- Since it is out of scope for this study to develop and test multiple models, it is important that a **well-performing model structure for this dataset is available**.
- As will be described in the next section, parts of the dataset will be relabelled for personalization. This means that an evaluator should be able to **recognize activities** from the data.
- As will be described in-depth in the next section, personalization will be achieved through combining activities. However, these **activities should be similar** to be combined logically. Therefore, the dataset requires at least twice two activities that can be confused with each other.

While there are several different HAR datasets containing 3D joints, most do not comply with the requirements. There is a high number of HAR datasets containing a small number of samples, i.e. less than 10, per person [119]–[125]. Other datasets are imbalanced [126], use 2D data [127] and/or contain low quality joint estimations [126], [128]. Next to that, some datasets focus on interactions, rather than individual actions [129]. Lastly, some datasets focus on gestures [130], which may be too difficult to recognize while labelling.

The NTU RGB+D 120 dataset, however, complies with all requirements and contains at least two people for which 40 samples per activity per person were gathered [29]. It contains RGB videos, depth sequences, 3D skeleton data (consisting of 25 joints) and infrared camera footage, all captured by three Kinect V2 cameras. In this study, only the skeleton data is used. The dataset contains 120 classes, including daily actions, health-related actions and

interactions. The activities are performed by 106 subjects of different ages (10 to 58), heights (1.3m to 1.9m) and from 15 different countries. Additionally, the activities are captured with 96 different backgrounds and 155 different viewpoints.

Since this study focuses on personalization, the data must be personalized. The NTU RGB+D 120 dataset does not deliberately include personalization. Therefore, a persona was designed with specific restraints, for which the data was relabelled by annotators. This is similar to real-life situations, where a personal restraint may be applicable. The type of personalization was equal to the combined dataset from the toy problem, where the inputs of two classes map towards one output class. To test this setup, a cross-subject evaluation was performed for two subjects.

Class Selection Since it was not necessary to use all 120 classes for this study, and decreasing the number of classes increases the clarity of the analysis, only six classes were selected, along with an *other* class. An exploratory test showed which classes could logically be combined, according to the following requirements:

- Relatively **easy to recognize**. Some activities were difficult to classify, because of their minimal movements, or because it was difficult to create a video with a clear skeleton due to the joints getting mixed up.
- **Two classes that look similarly** to each other, so that they can logically be combined. For example, butt kicks and clapping cannot be logically combined because their input is too different.
- **Two combinations that do not look similarly** to each other. The goal is not to confuse both combination examples with each other, so these two combinations should not be too similar.

From these requirements, *yawn* and *sniff/smell* were manually picked because they are similar in terms of movement and relatively well recognizable. Additionally, *staple book* and *cutting paper* were also manually picked because they look similar but are quite different from *yawn* and *sniff/smell*. Next to that, two classes were randomly selected out of the remaining classes as a control group, for which no labels were changed. It is important to note that the first 60 classes, i.e. A001 to A060, were performed by one group of subjects and the last 60 classes, i.e. A061 to A120, were performed by a different group of subjects, therefore the two groups contain no overlap. Since this study performs its evaluation across subjects, all activities must be performed by the same person. Therefore, only one part of the dataset, namely classes A061 to A120, was considered for the random choice in control classes. Additionally, since this study does not include interactions, all classes containing interactions, i.e. A106 to A120, were disregarded. The two control classes were randomly selected from the remaining classes which resulted in the addition of *cutting nails* and *put on headphones* as classes. Moreover, an *other* class was used, which consists of data from all

other classes between A061 and A105, excluding the previously chosen classes. To ensure the dataset remains balanced, the same number of samples for the *other* class was used as were available for the other classes by random undersampling. In conclusion, the following seven classes were used in this experiment: *yawn*, *sniff/smell*, *staple book*, *cutting paper*, *cutting nails*, *put on headphones* and *other*.

The experiments were performed on two subjects to decrease the influence of chance.[23] Two subjects in the dataset contain a considerably higher number of samples than the other subjects, namely 420 samples, as opposed to generally 42 to 126 samples and at most 336 samples. Due to this difference, the subjects with 420 samples, called P008 and P041, were chosen for this experiment.

Personalized Labelling As mentioned previously, the classes were combined, similarly to the toy problem. However, instead of combining the classes automatically, they were relabelled by annotators to find out whether the combination of classes is logical. If the combination of classes is illogical, it is expected to yield many labels with *I don't know* or *other*. Two annotators were assigned to provide feedback on the samples from one subject each. The two annotators consisted of a technical medicine student and an embedded software engineer, who had no inside information on the experiments that would be performed with the relabelled data. The technical medicine student was tasked with relabelling the data for P008 and the embedded software engineer for P041.

To visualize the samples, the 3D joint information was converted into a video of a skeleton performing an activity. Figure 11 shows an example of a frame from such a video.

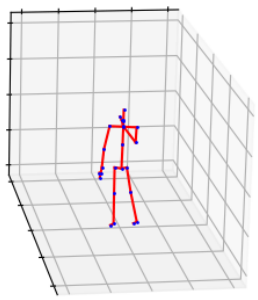


Figure 11: A frame from a skeleton video of a person performing the action *sniff/smell*.

[23] Unfortunately, due to time constraints, it was not possible to evaluate on more subjects.

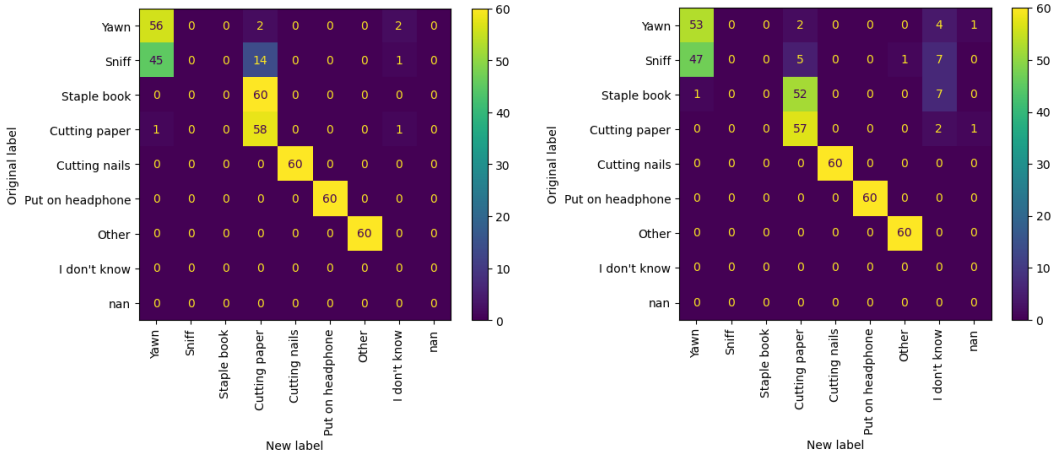
The annotators were asked to label each sample with the following options:

1. Yawn
2. Sniff/smell
3. Staple book
4. Cutting paper
5. I don't know
6. Other

As described in Section 1.3, handling noisy labels is out of scope for this study. To limit the chances of yielding noisy labels, the unchanged classes, i.e. *cutting nails* and *put on headphones* were excluded from the options. Additionally, *I don't know* was added as an option, which could be chosen if the activity could not be interpreted. This occurred, for example, when no feasible skeleton could be made from the data, or when the activity was simply too much alike multiple classes. All samples labelled *I don't know* were discarded during the experiments. The annotators were asked to label the data according to a persona who has two characteristics:

- This person has lost their sense of smell and therefore has not sniffed/smelled during the data acquisition.
- This person has no stapler at home so cannot staple books.

The distribution of the relabelled samples is shown in Figure 12. The labels for *cutting nails* and *put on headphones* remained the same, one additional sample was labelled *other* for P041, most samples from *sniff* were relabelled as *yawn* and most samples from *staple book* were relabelled *cutting paper*. Only a minor number of samples were labelled differently: 17 samples for P008 and 8 samples for P041 received a different label than expected. Since this is only a minor part of the samples, it most likely did not have a great influence on the results. Furthermore, 4 samples for P008 and 18 samples for P041 were unrecognizable and labelled as *I don't know*. For P041, 2 samples received the label *nan* because the annotations were missing. All samples labelled *nan* or *I don't know* were disregarded. The precise labelling instructions, along with observations and a discussion of the labelling task can be found in Appendix B.



(a) P008 (b) P041

Figure 12: New labels that the annotators gave for the two subjects.

To include LF, the annotators would be asked a question with just a few options, instead of all classes. However, to decrease the labelling effort in this study, the labels were given without LF and later translated. For example, if the model predicts *yawn* and $e = 1$, the question can be translated to "Is this *yawn* or something else?". If the label given by the annotator was not *yawn*, the answer is considered *something else*. In this way, LF was added during the processing of the data, unlike it would in a real-life scenario.

After relabelling the samples, they were split into a train and test set. Because the experiments use relatively few samples, five-fold cross-validation is used. The samples are randomly split up into five folds, using 20% of the data for testing and 80% of the data as feedback set F.[24] The feedback set F was, in turn, split into train set T, with 66% of the data and comparison set C, with 33% of the data. Lastly, train set T was again split up into 33% validation and 66% training data. The data from the other subjects, consisting of 6300 samples, was split up into a training (66%) and validation (33%) set for the general model.

General and Personal Model The same model was used as a general and personal model, namely ST-GCN++ (Spatio-Temporal Graph Convolutional Network++) [131]. ST-GCN++ yields state-of-the-art performance on multiple datasets, including the NTU RGB+D 120 dataset. Additionally, it is also more computationally efficient than most other algorithms with similar performance on the NTU RGB+D 120 dataset. The implementation of ST-GCN++ from MMAction2 was used in this study [132].

The settings for training the models were unchanged from MMAction2, with the main settings as follows: a batch size of 16, accuracy as validation metric and a maximum of 16 epochs. The general and personal models were trained with an SDG optimizer with a momentum of 0.9 and weight decay of 0.0005. The general model was trained with a learning

[24] Please note that, due to samples labelled as *I don't know* and *nan*, the total number of samples differed slightly between the two subjects: P008 consisted of 416 samples and P041 of 398 samples.

rate of 0.1. The personal models inherited the weights from the general model, after which they are fine-tuned, using a reduced learning rate, as advised by MMAction2 [133]. This reduced learning rate was set to 0.05.

The personal and general models were both trained and evaluated on the two subjects, P008 and P041. Two separate general models were trained, one on all data from the relevant classes from all subjects excluding P008, and one excluding P041. By training two separate general models, the number of samples to train a general model was maximized. Afterwards, the personal model inherited the weights of the general model and continued training on personal samples with a reduced learning rate.

Similarly to the toy problem, all models were trained with a varying number of training samples: 20, 40, 60, 80, 100, 120 and 140 samples to get better insights into the training process.

5.3 Evaluation Measures

To evaluate the methods, different evaluation metrics and visualizations were used, which will be explained below.

5.3.1 Metrics

The four metrics used in this study are accuracy, precision, recall and F1-score. The accuracy was generally used to evaluate the overall performance of the models. While accuracy is not always an advised metric, especially in an imbalanced dataset, it is a reasonable metric to use in this study, since the classes are relatively balanced [134]. Apart from that, precision, recall and F1-score were used to evaluate models per class.

5.3.2 Visualizations

Different visualizations, including figures and tables, were used to evaluate the models. All visualizations used to describe the results will be explained below.

Accuracy Table An *accuracy table* simply depicts the accuracy of multiple models, optionally for different values of $|T|$. The accuracy is measured by averaging the accuracy over five-fold cross-validation. It shows the accuracy for the three datasets in the toy problem and for the two subjects for HAR.

Class-Specific Performance Table To get more insights into the performance per class, a *class-specific performance table* measures the performance for multiple models for each class. Precision, recall and F1-score are all denoted in this table, and calculated by averaging the results from five-fold cross-validation.

Confusion Matrix A *confusion matrix* depicts a summary of the predictions of all samples. It shows the predicted class from each sample against the actual class. Through this, it becomes apparent which classes are often confused by each other. Additionally, a *confusion matrix* can help to find for which classes a model lacks performance.

Model Choices Visualization To show which choices were made by ND (Novelty Detection) and DCS (Dynamic Classifier Selection), a *model choices visualization* can be made. This visualization shows which model was chosen for each test sample and the reason for that choice. All samples are first classified as *novel* or *non-novel* by the ND component, after which DCS decides on the classifier to use for the non-novel samples. This results in three classification options: (1) the general model's prediction, because the sample was classified as novel, (2) the general model's prediction, because it was chosen through DCS, and (3) the personal model's prediction because it was chosen through DCS. The goal of the *model choices visualization* is to show which trends are seen in the choices of ND and DCS. This can help in the discussion on whether ND and DCS function as expected. To prevent over-complication, this visualization is made with one-fold cross-validation.

It can be made with different feedback set sizes, which fulfil different goals. Specifically, a visualization for a small feedback set, and thus also a small comparison set, can help evaluate the contribution of novelty detection since this is most relevant for small feedback sets. When the comparison set is relatively small, the chance of encountering novel samples in the comparison set is high, which increases the focus on ND. On the other hand, when the comparison set is relatively big, there are more non-novel samples, and thus DCS is used more. Therefore, a visualization with a small feedback set increases the focus on ND, whereas a bigger feedback set increases the focus on DCS.

Sampling Choices Visualization To better understand how sampling strategies work, a *sampling choices visualization* can be made. It visualizes a dataset and which samples were chosen by an AL sampling strategy. It shows the initial pool of samples which was randomly chosen, along with the chosen samples by the strategies. This visualization was only performed with one-fold cross-validation to prevent over-complication.

Sample Size Analysis Plot Since it is important that the system can function with a minimal number of feedback samples, as explained in Section 1, it is important to get better insights into how quickly a model learns. To achieve this, a *sample size analysis plot* can be made to visualize the accuracy over different training set sizes. The range of $|T|$ differs per experiment, as can be seen below.

Dataset	Experiment	Range of $\|T\|$
Toy problem	Comparing the personal model with and without the framework	4 to 100
	Comparing LF techniques	4 to 100
	Comparing AL sampling strategies	4 to 15
HAR	Comparing the personal model with and without the framework	20 to 140
	Comparing LF techniques	20 to 140
	Comparing AL sampling strategies	20 to 80

Table 2: Range of $|T|$ used in each experiment, to visualize the performance over $|T|$.

The visualization shows a mean accuracy over five-fold cross-validation runs. Additionally, the minimum and maximum accuracy reached in those folds is visualized, if this was possible without yielding cluttered results. The minimum and maximum accuracy can help to get an indication of the variation of the performance.

Label Request Analysis Plot As explained in Section 5.1.2, the LF techniques were tested for different values of e. To visualize this, a *label request analysis plot* can be made, which plots the accuracy over the number of requested labels. This plot is similar to the *sample size analysis plot*, except it measures the performance for different values of e, instead of measuring the performance for different values of $|T|$. This visualization helps to find the optimal value of e and to compare the LF techniques in different situations.

Prediction Comparison Table To find out for which classes the framework increased the number of correct predictions the most, the predictions of the personal model with the framework can be compared to the predictions of the personal model without the framework. For each sample in each cross-validation fold, the prediction of the personal with and without the framework was compared to the real label. This resulted in one of four options: (1) both models were correct, (2) only the model without the framework was correct, (3) only the model with the framework was correct and (4) both models were incorrect. By counting how many times each option occurred per class, a *prediction comparison table* can be made. The goal of this is to understand where the framework helps and lacks. This visualization was made with the maximum number of training samples for each dataset, i.e. $|T| = 100$ for the toy problem datasets and $|T| = 140$ for HAR.

Convergence Rate Table A *convergence rate table* shows how quickly each model converges in terms of the number of epochs required to yield the maximum performance. While this study mainly focuses on yielding a high accuracy, recall, precision and F1-score, the

convergence rate could be useful in determining a preferred model if all models perform similarly in the previously mentioned metrics.

To yield this table, each model was run for the maximum number of epochs, i.e. 50 for the toy problem and 16 for HAR, after which the epoch with the lowest validation loss was chosen. The table reports a mean and standard deviation from the five cross-validation runs. Additionally, this evaluation was performed with the maximum number of samples for each dataset, i.e. $|T| = 100$ for the toy problem datasets and $|T| = 140$ for HAR.

Precision-Recall Plot A *precision-recall plot* shows the precision and recall values at different thresholds. This can help to compare the performance of models and find the right threshold. It is normally used in binary classification, although the micro-average can be calculated to allow multi-class classification. The precision-recall plots were generated with the maximum number of training samples, i.e. $|T| = 100$ for the toy problem and $|T| = 140$ for HAR. Additionally, the predictions and labels from five cross-validation folds were combined. The AP (Average Precision) was also calculated, which indicates the area under the curve, where a higher AP is desired.

6 Results

In this chapter, the outcomes of the experiments are presented. The results are organised into two subsections, each describing the outcomes on a specific dataset. Section 6.1 describes the findings from the toy problem, followed by Section 6.2, which describes the insights gathered from the HAR (Human Activity Recognition) dataset. Each dataset analysis covers three topics: framework performance, various LF (limited feedback) techniques and two AL (Active Learning) sampling strategies.

The primary objective of the analysis of framework performance is to show the impact of the framework by comparing the performance with and without the framework. Additionally, the toy problem will be used to further explain the functioning of the framework and its components, ND (Novelty Detection) and DCS (Dynamic Classifier Selection), through visualizations. During the evaluation of LF techniques, emphasis is placed on identifying the best-performing LF techniques and the trade-off between the minimum number of labels that need to be requested from the annotator and the maximum performance. The AL results focus on determining whether entropy-based sampling improves the performance over random sampling. Additionally, the minimum number of samples required for each sampling strategy to yield an optimal performance will be discussed. By evaluating the framework, LF techniques and AL strategies on both the toy problem and the HAR dataset, the difference in performance of the techniques can be shown across two diverse datasets - one low-dimensional with a small model and one high-dimensional dataset with a complex model.

6.1 Toy Problem

6.1.1 Framework

First, the results of the personal and general models will be presented separately. This provides the baseline performance for the personal model without the use of the framework. Second, the functioning of ND and DCS will be explained with visualizations, which will help in the further understanding and discussion of the framework. Third, the framework will be added to the personal model, so that the difference in performance of the personal model with and without its integration can be compared.

As mentioned before, this section starts with an evaluation of the personal model without the framework and the general model. The models were tested on the drift, unexplored and combined datasets, as explained in Section 5.2.1. Table 3 shows an accuracy table for the general model and the personal model, without framework, trained on four samples to show well well they perform after training on a small set of samples. The personal model yields an increase in accuracy after training on four samples for all three datasets. Specifically, the personal model yields the smallest improvement after training on four samples on the unexplored dataset, while it yields the biggest improvement on the drift and combined

datasets. This shows that after training on only four samples, the personal model can improve upon the general model for all three toy problem datasets, although the rate of improvement differs.

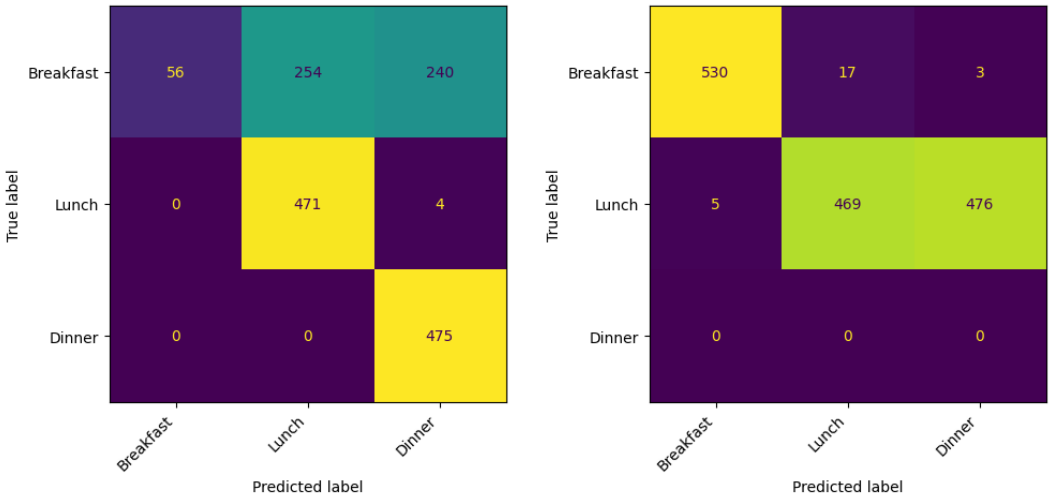
However, this increased performance is not yet the highest performance that the personal model can reach. To show how the performance increases over a greater number of samples, Table 3 also reports the accuracy after training on 100 samples. This shows that the personal model can improve further for all three datasets. Additionally, the personal model yields a higher accuracy for the drift and combined datasets than for the unexplored dataset.

	General Model	Personal Model	
		$\|T\| = 4$	$\|T\| = 100$
Drift	0.668	0.849	0.973
Unexplored	0.667	0.749	0.849
Combined	0.666	0.801	0.964

Table 3: Mean accuracy with five-fold cross-validation of the general and personal model with training set T containing 4 and 100 personal samples on the drift, unexplored and combined datasets.

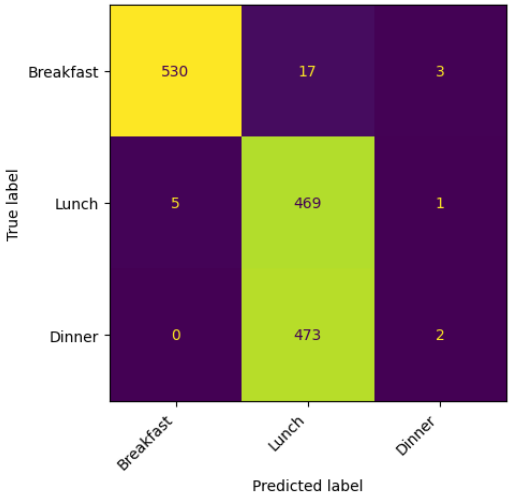
As explained in Section 4.4, the framework aims to use the expertise of the personal and general model to optimize the performance. To know where the framework can further increase the performance, it is first important to know where the general model lacks performance. This knowledge can assist in understanding the framework's functioning later on.[25] To identify the classes for which the general model lacks performance, a confusion matrix is shown in Figure 13. It shows that the general model lacks performance for one class in each dataset, although the class differs. For the drift dataset, most of the incorrect predictions stem from the *breakfast* class, which has shifted into the regions where the *dinner* and *lunch* classes were previously. Similarly, for the combined dataset, most changed samples from *lunch* were classified as *dinner*, and for the unexplored dataset, most samples from *dinner* were classified as *lunch*.

[25]The goal of the toy problem datasets was to artificially create situations in which the general model does not perform optimally, although not disastrous either. Table 3 shows that the performance of the general model is similar for the three datasets, with approximately 33% incorrect predictions. Therefore, this performance matches the design goal of the toy problem datasets.



(a) Drift dataset

(b) Combined dataset



(c) Unexplored dataset

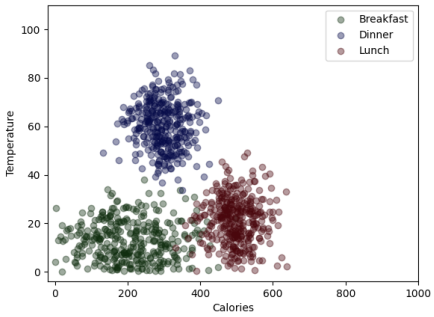
Figure 13: Confusion matrix showing the predictions and real labels of the general model, across all five cross-validation sets. The subfigures for the datasets (a) drift, (b) combined and (c) unexplored show which class was mainly predicted incorrectly by the general model.

Before discussing the performance of the framework, it is essential to discuss the functioning of the ND and DCS components, as explained in Section 4.3 and 4.4. These components will be visualized using a model choices visualization to provide a clearer understanding of how the framework operates. Figure 14c and 14d show which model was used for the classification of a test sample from the unexplored dataset and whether the choice was based on ND or DCS. The general and unexplored datasets, as previously explained in Section 5.2.1, are repeated in Figure 14a and 14b for easy reference.

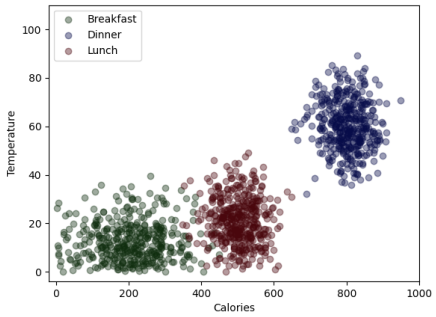
Figure 14c shows the ND and DCS choices with $|F| = 6$, i.e. $|C| = 2$. In this figure, both samples in C are from the *dinner* class. The ND component classifies all test samples in the same direction as these samples as non-novel, which are mostly samples from the *dinner*

class. However, also some samples from *lunch* are classified as non-novel. The reason for this will be explained in Section 7.1.1. Additionally, all non-novel samples are classified by the personal model.

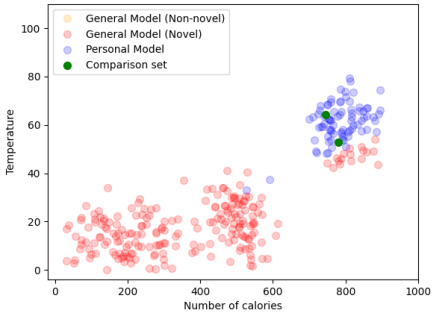
Figure 14d shows the results of ND and DCS with $|C| = 20$. No samples are classified as novel in this figure. Because the samples in comparison set C are spread sufficiently, all test samples are relatively similar to the samples in C and are therefore classified as non-novel. Additionally, the personal model is chosen for most samples, and all samples in the *dinner* class, while the general model is chosen for some of the samples in the *breakfast* and *lunch* classes. The reason why each model is preferred for each class will be further explained in Section 7.1.1.



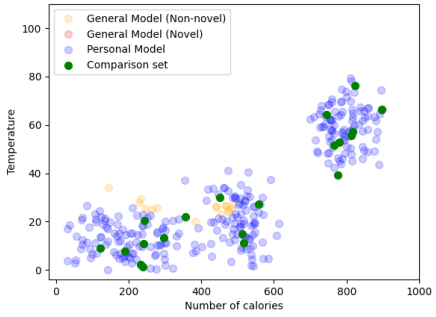
(a) General dataset



(b) Unexplored dataset



(c) The samples classified as novel and non-novel, including the chosen classifier by DCS, of the unexplored dataset with $|C| = 2$ for one cross-validation set.



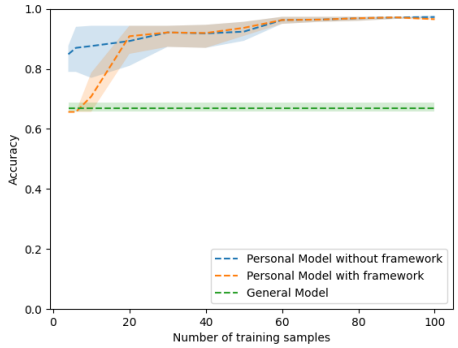
(d) The classifier selected for each test sample in the unexplored dataset, along with the samples in comparison set C, with $|C| = 20$ for one cross-validation set.

Figure 14: Visualizations of the components of the framework. The (a) general dataset and (b) unexplored dataset are shown as reference, whereas (c) and (d) show the chosen classifier through ND and DCS, based on comparison set C.

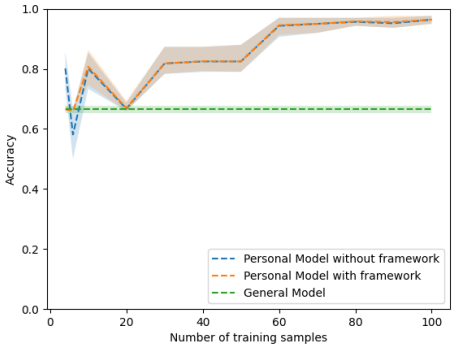
Now that a better understanding of the functioning of ND and DCS has been achieved, the framework can be applied to the personal model. The following analysis aims to show the performance difference between the personal model with and without the framework through a sample size analysis plot. Figure 15 shows the mean accuracy of the personal model with

and without the framework in comparison to the general model. For all three datasets, the performance of the personal model is similar regardless of the use of the framework.[26]

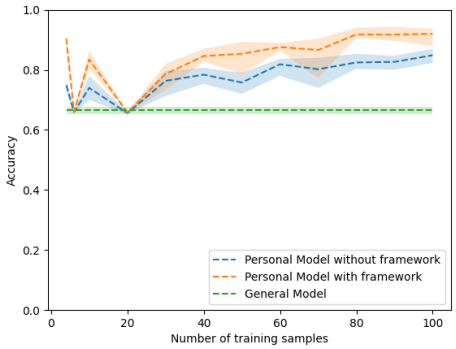
As was shown in Table 3, the personal model can improve after only 4 training samples. However, Figure 15 also shows a high variation in performance, especially when the training set is small. Fortunately, the performance steadily increases for all three datasets as the number of training samples grows. Please note that the models were only run with five-fold cross-validation, so it is not possible to indicate whether the variation in accuracy is because of outliers or normal behaviour.[27]



(a) Drift dataset



(b) Combined dataset



(c) Unexplored dataset

Figure 15: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) of the personal model without the framework, the personal model with the framework and the general model on 5 cross-validation sets for the (a) drift, (b) combined and (c) unexplored datasets.

[26] While it was expected that the framework would yield a high performance increase in the toy problem datasets, the reason for the absence of a difference in performance for this dataset makes sense in retrospect and will be explained in Section 7.1.1.

[27] Unfortunately, it was not possible to increase the number of cross-validation folds due to time restraints.

To get a better view of the performance of the models, the predictions of the personal model without the framework were compared with the predictions of the personal model with the framework with $|T| = 100$ using a prediction comparison table.[28] This can give an insight into the classes where the framework increased the number of correct predictions the most.

Table 4a shows that there are only minor differences between both models for the drift dataset since most samples are either predicted correctly by both models or incorrectly by both models. Table 4b shows that the framework always predicts the same class as the personal model for the combined dataset.[29]

Table 4c shows that the framework corrected 22.1% of the samples of the *lunch* class that were predicted incorrectly by the personal model for the unexplored dataset. This is also the main cause of the slight increase in performance seen in Figure 15c. Additionally, little to no difference was seen between the predictions of the *breakfast* and *dinner* classes.

Please note that the following sections on the toy problem were conducted with the framework. Only the results with the framework will be shown in this chapter to avoid cluttered results. The results without the framework can be found in Appendix C.1.

[28] Table 4 shows the results in terms of percentages. The raw data can be found in Table 17 in Appendix C.2.

[29] The fact that the framework always predicts the same class as the personal model does not necessarily mean it always chooses the personal model. The general model might have been chosen for samples where the general model yielded the same prediction as the personal model. This observation only indicates that the framework never used a general model's prediction which is different from the personal model.

		Personal Model					
		Breakfast		Lunch		Dinner	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Personal model with framework	Correct	0.936	0.000	0.979	0.000	1.000	0.000
	Incorrect	0.018	0.045	0.000	0.021	0.000	0.000

(a) Drift

		Personal Model			
		Breakfast		Lunch	
		Correct	Incorrect	Correct	Incorrect
Personal model with framework	Correct	0.945	0.000	0.958	0.000
	Incorrect	0.000	0.055	0.000	0.042

(b) Combined[30]

		Personal Model					
		Breakfast		Lunch		Dinner	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Personal model with framework	Correct	0.973	0.000	0.695	0.221	0.916	0.000
	Incorrect	0.009	0.018	0.000	0.084	0.000	0.084

(c) Unexplored

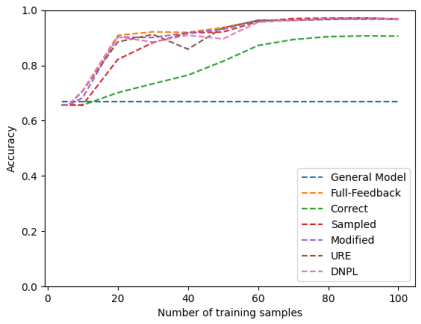
Table 4: Portion of the correctly and incorrectly predicted samples from the personal model and personal model with the framework for the (a) drift, (b) combined and (c) unexplored datasets.

[30]Please note that the test set of the combined dataset does not contain any samples from the *dinner* class, which is therefore not included in this table.

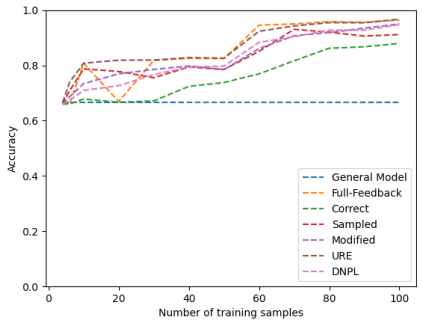
6.1.2 Limited Feedback

In this section, the performance of five LF techniques (*Correct*, *Sampled*, *Modified*, *URE* and *DNPL*) will be presented and compared to a *Full-Feedback* model.[31] As explained in 5.1.2, these LF techniques receive a soft label if the correct prediction is not among the e highest predictions, where e represents the number of labels presented to the annotator. Since the toy problem consists of only 3 classes, $e = 2$ would yield a hard label for every sample. Therefore, only $e = 1$ is tested for the toy problem.

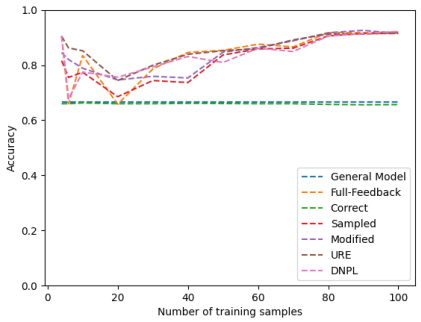
To get better insights into the training process, Figure 16 shows a sample size analysis plot for 4 to 100 samples.[32]



(a) Drift dataset



(b) Combined dataset



(c) Unexplored dataset

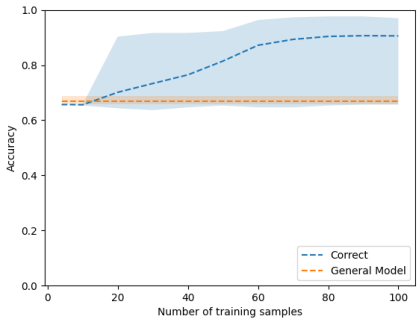
Figure 16: Mean accuracy of the LF techniques *Correct*, *Sampled*, *Modified*, *URE* and *DNPL* in comparison to the *Full-Feedback* model and general model with five-fold cross-validation for the (a) drift, (b) combined and (c) unexplored datasets.

[31] This *Full-Feedback* model is the same as the personal model in the previous section, which also received hard labels for every sample. Technically, any model can be used as a personal model, therefore the name was changed between sections.

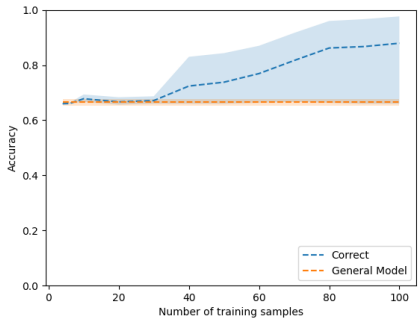
[32] The variation in performance was not included, since this would lead to cluttered results, but can be found for each method separately in Appendix C.3.

Almost all techniques, i.e. *Sampled*, *Modified*, *URE* and *DNPL*, perform similarly on the three datasets. Interestingly, these techniques also perform similarly to the *Full-Feedback* model, even though the *Full-Feedback* model receives a hard label for every sample. Additionally, the *Correct* technique yields a lower mean accuracy than all other techniques in all datasets. Notably, it fails to improve upon the performance of the general model in all five cross-validation sets for the unexplored dataset.

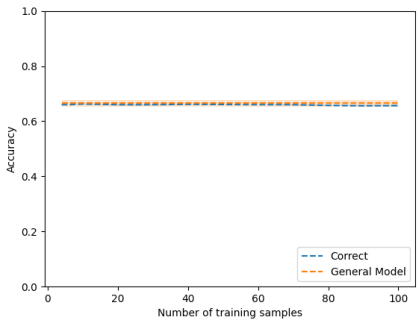
While the variation could not be depicted in Figure 16 due to clutteredness, the variation in performance can give a further indication of how well a technique performs. The *Correct* technique yields a high variation in performance, which means that only evaluating a mean value may not be the most informative way to evaluate its performance. Therefore, Figure 17 shows the variation in performance in a sample size analysis plot. While five-fold cross-validation is not enough to prove a significant difference in variation, it can help to better understand the performance than simply viewing the mean.



(a) Drift dataset



(b) Combined dataset



(c) Unexplored dataset

Figure 17: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) for the drift, unexplored and combined datasets for the *Correct* technique with five-fold cross-validation.

The accuracy for the drift and combined dataset varies between the performance of the general model, i.e. around 66% accuracy, and nearly optimal performance, i.e. 97% accuracy, across the five cross-validation runs. The *Correct* technique can converge to an accuracy of 97% after just 20 training samples in some runs, but it is also possible that the performance does not converge at all, and fails to improve upon the general model after 100 samples. Additionally, the *Correct* technique does not improve upon the general model in any of the cross-validation runs for the unexplored dataset. These insights are important because they show that the slow convergence in the mean value does not necessarily mean that the *Correct* technique always converges slowly, but rather that it is caused by averaging the results from the five runs. Since the variation for the other techniques is much lower, it is less relevant to discuss here. Therefore, the variation plots for the other techniques can be found in Appendix C.3.

Because the performance of most LF techniques is similar in terms of accuracy, it is interesting to see if they differ in convergence rate and if they tend to over- or underfit. If all LF techniques yield a similar accuracy, but one would yield that performance substantially quicker, this may be a reason to prefer that technique. Table 5 shows this in a convergence rate table with $|T| = 100$. The number of epochs is relatively similar between all LF techniques. It also shows that most techniques train relatively slowly, in over 40 epochs, or even up to the maximum of 50 epochs. In general, there is not enough difference between the LF techniques to claim that some techniques converge quicker than others.

	Full-Feedback	*Correct*	*Sampled*	*Modified*	*URE*	*DNPL*
Drift	49.8	48.8	50.0	44.2	46.6	45.6
Unexplored	49.6	38.2	42.0	42.4	50.0	50.0
Combined	47.6	50.0	43.6	47.6	48.2	50.0

Table 5: Mean number of epochs required to yield the lowest validation loss, i.e. the best-performing epoch, with five-fold cross-validation of the LF techniques on the drift, unexplored and combined datasets.

6.1.3 Active Learning

In this section, two AL strategies will be compared: entropy-based sampling and random sampling. Specifically, this section aims to visualize entropy-based and random sampling, to determine the number of samples required to yield an optimal performance and to compare the performance between entropy-based and random sampling.

As explained in Chapter 5, these experiments were only conducted with one LF strategy to prevent over-complicating the analysis. Given the similar performance between *Sampled*, *Modified*, *URE* and *DNPL*, a conclusion on the best-performing strategy was not possible.

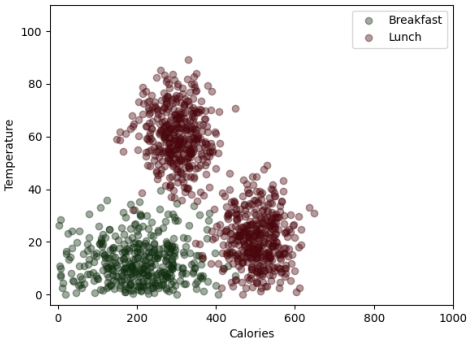
Therefore, one of these four strategies was chosen for further experiments with AL for the toy problem, namely the *URE*.

The training process with AL resembles the training process of the previous experiments, meaning that the number of samples was increased incrementally from 4 to 100 samples. However, instead of training on all of these samples, the most relevant sample is selected out of each pool as described in Section 5.1.3, resulting in a range from 4 to 15 training samples.

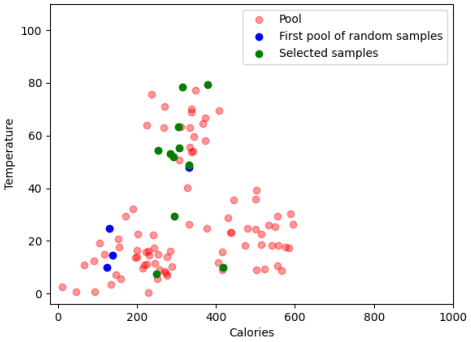
First, entropy-based and random sampling will be visualized using a sampling choices visualization on the combined dataset to increase the understanding of the strategies.[33] Figure 18b shows that entropy-based sampling primarily selects the samples from the *dinner* class, which are the samples that have been changed from the general data. Additionally, samples near the decision boundary between *lunch* and *breakfast* have been chosen. This shows that the model focuses on the samples that have been changed, as opposed to samples that are similar to what it has already learned. Random sampling, on the other hand, selects samples randomly, as can be seen in Figure 18c.

After visualizing the sampling strategies, the next step is to find the number of samples required to yield optimal performance, as well as compare the performance between entropy-based and random sampling, which was done using a sample size analysis plot.

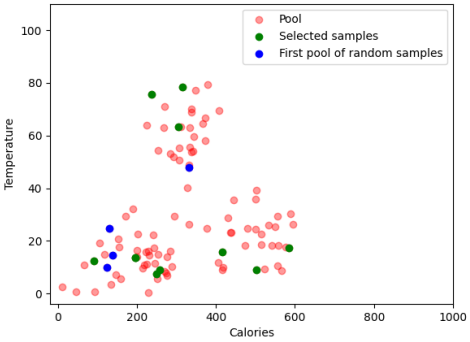
[33] Figure 18a can be used as a reference for the combined dataset.



(a) Combined dataset



(b) Entropy

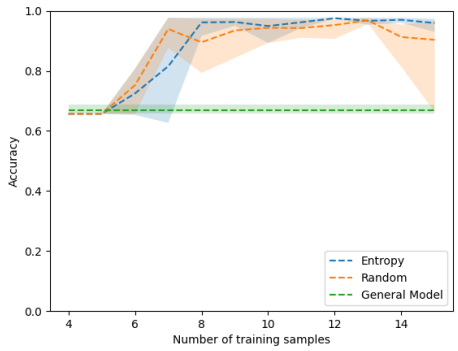


(c) Random

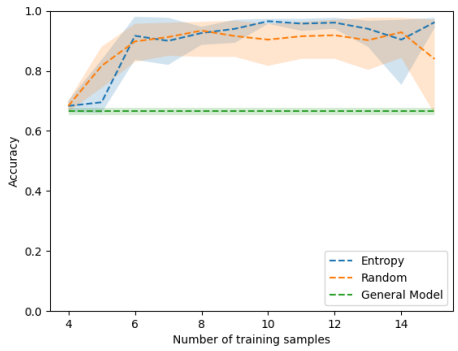
Figure 18: The samples selected using (b) entropy-based and (c) random sampling, based on the combined dataset, which is shown in (a) as a reference. The first four samples are selected randomly as a basis, after which the remaining eleven samples are selected using the sampling strategy.

It shows that both sampling strategies perform similarly for the three toy problem datasets. The mean accuracy converges after 7 or 8 training samples for both sampling strategies for the drift dataset. Similarly, both strategies converge after 6 samples for the combined dataset. Lastly, the mean accuracy for entropy-based sampling for the unexplored dataset seems to decrease slightly over the number of samples. In all three datasets, the variation in performance varies between the general model's accuracy, i.e. around 66%, and a nearly optimal accuracy, i.e. around 97%. Due to this variation, no conclusions can be made on the number of samples required to yield an optimal performance. Additionally, the performance is very similar between random and entropy-based sampling in all three datasets. The reason for this will be explained in Section 7.1.3.

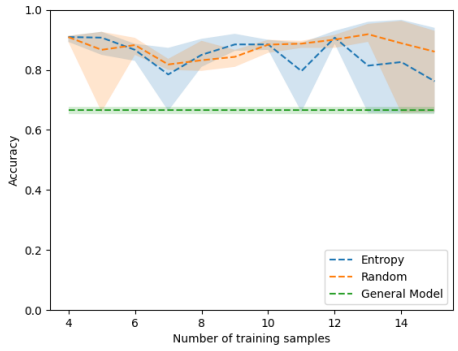
This concludes the results of the experiments for the toy problem datasets. The next section will show the results of the experiments on the HAR dataset.



(a) Drift dataset



(b) Combined dataset



(c) Unexplored dataset

Figure 19: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) of the *URE* with the framework with entropy-based sampling and random sampling in comparison to the general model with five-fold cross-validation for the (a) drift, (b) combined and (c) unexplored datasets.

6.2 Human Activity Recognition

6.2.1 Framework

This section contains the evaluation of the personal and general models and the framework on the HAR dataset. It starts with the results of the personal and general models individually, including a comparison between the performance of the models on unpersonalized and personalized labels. With this, insight is given into how much the performance is affected by the personalization of labels. Lastly, the framework is added to the personal model to see how the performance is affected by the introduction of it.

As mentioned previously, the performance of the general and personal models will be discussed first. Table 6 shows an accuracy table for the general model and the personal model with 20 training samples to see how well the personal model improves upon the general model with few samples. This evaluation is performed using personalized labels, i.e. with restrictions on the classes. This results in an increase in accuracy by 0.050 for P008 and

a decrease by 0.229 for P041. To find out if this decreased performance also occurs when training on more samples, the personal models for P008 and P041 were also trained with 140 samples. Table 6 shows that the performance for both subjects decreases in comparison to the personal models trained on 20 samples. The reason for the decreased performance, after 20 and 140 training samples, will be discussed in Section 7.1.1, along with the limitations this brings to the other experiments.

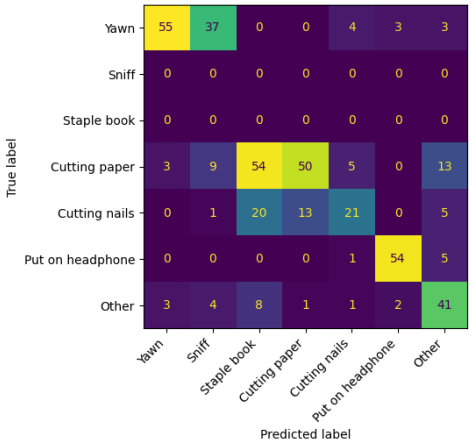
	General Model	Personal Model	
		$\|T\| = 20$	$\|T\| = 140$
P008	0.531 ± 0.027	0.581 ± 0.066	0.544 ± 0.157
P041	0.611 ± 0.034	0.387 ± 0.206	0.302 ± 0.069

Table 6: Mean accuracy and standard deviation of the general and personal model trained on 20 and 140 samples with personalized labels.

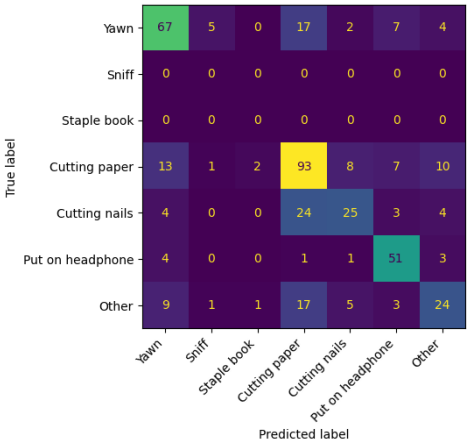
To get a better indication of the performance of the models, it is important to view how well they can classify each class. Figure 20 shows four confusion matrices that contain the predictions of the general and personal models for each person over five-fold cross-validation. Additionally, a class-specific performance table was made for the general and personal models for each subject, as can be seen in Table 7. This experiment was specifically performed for $|T| = 140$, to see where the personal models excel and lack performance after training on the maximum number of samples.

The general model predicts most classes well for P041, with precision and recall above 0.7 for each class, with the exception of the two personalized classes: *yawn* and *cutting paper*. Specifically, it predicts part of the *yawn* samples as *sniff*, and part of the *cutting paper* samples as *staple book*, which results in low recall. Additionally, it yields the most remaining mistakes for *other*. The general model performs worse for P008, where it yields the lowest F1-score for *cutting nails*. Additionally, similarly to P041, the general model yields the lowest recall scores for *yawn* and *cutting paper* due to them being personalized.

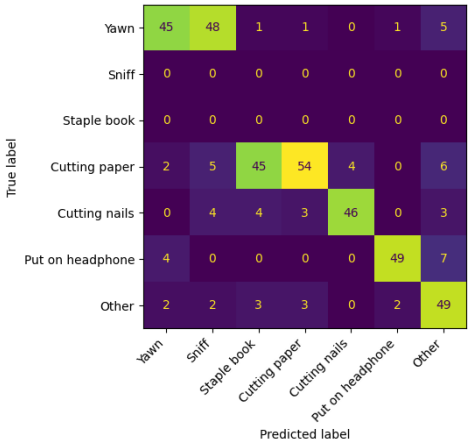
The personal models yield no incorrect predictions of *sniff* and *staple book*, which indicates that they have correctly learned to refrain from classifying samples as these classes. However, the precision decreases for all classes for both subjects. Additionally, the recall decreases for *cutting nails*, *put on headphones* and *other* for both P008 and P041, which are the unchanged classes. This decrease can also be seen in Figure 20. For example, relatively many samples, i.e. 17 for P008 and 25 for P041, from *yawn* are classified as *cutting paper* by the personal model while this mistake was not made often by the general model. As mentioned previously, this decreased performance will be discussed further in Section 7.1.1.



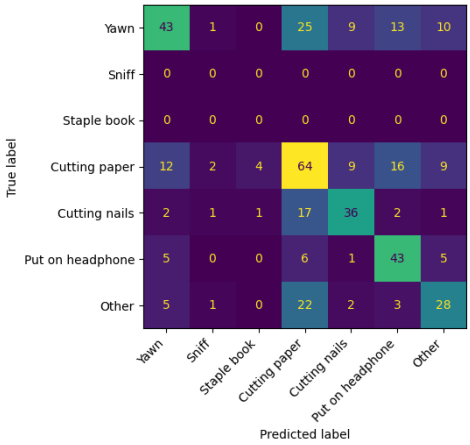
(a) General model, P008



(b) Personal model, P008



(c) General model, P041



(d) Personal model, P041

Figure 20: Confusion matrix showing the predictions and real labels of the general and personal models on the personalized data of P008 and P041.

	General			Personal		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Yawn	0.902	0.539	0.675	0.602	0.608	0.605
Cutting paper	0.781	0.373	0.505	0.522	0.694	0.596
Cutting nails	0.656	0.350	0.457	0.486	0.300	0.371
Put on headphones	0.915	0.900	0.908	0.607	0.567	0.586
Other	0.612	0.683	0.646	0.452	0.317	0.373

(a) P008

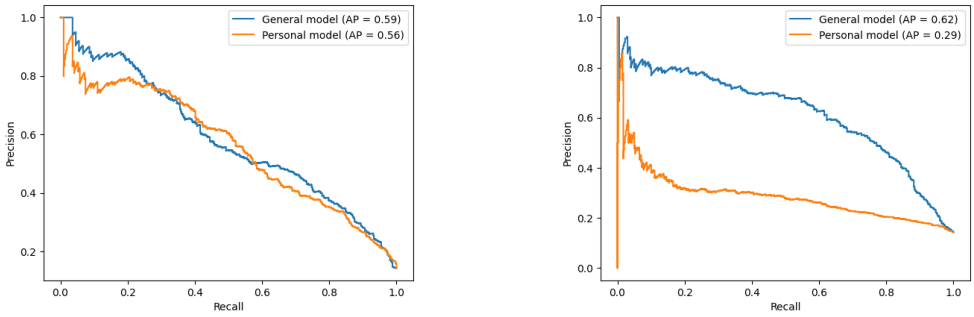
	General			Personal		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Yawn	0.849	0.446	0.584	0.373	0.218	0.275
Cutting paper	0.885	0.466	0.610	0.301	0.457	0.363
Cutting nails	0.920	0.767	0.836	0.255	0.217	0.234
Put on headphones	0.942	0.817	0.875	0.294	0.333	0.312
Other	0.700	0.803	0.748	0.273	0.197	0.299

(b) P041

Table 7: Precision, recall and F1-score in percentages for each class for the general and personal model for (a) P008 and (b) P041.[34]

Table 7 gives insights into class-specific performance. To better compare the models overall, a precision-recall plot was produced with $|T| = 140$. Again, personalized labels, i.e. with restrictions, were used for this evaluation. Figure 21 shows that the general and personal models yield similar curves and AP (Average Precision) values for P008. However, there is a difference for P041, where the general model yields a higher AP and precision is higher for all recall values from 0.1 to 0.9. This is an indication that the personal model performs similarly to the general model for P008, but it is outperformed by the general model for P041.

[34] *Sniff* and *staple book* were not included, because there are no samples from these classes due to the restrictions set.



(a) P008 (b) P041

Figure 21: Precision-recall plots of the general and personal model for (a) P008 and (b) P041.

Training a model on subject-specific data could increase its classification performance, even without knowledge on the personalized labels. To get a better indication of the impact of personalized labels, an experiment was done to evaluate the performance of the general and personal models on personalized and unpersonalized labels. In this way, the effect of the label adaption can be distinguished better, because the effect of subject-specific knowledge is minimized. Table 8 shows that the difference in performance between the general and personal models for the unpersonalized labels yields a 0.033 increase for P008 and a 0.064 decrease for P041. These differences give an indication of the effect of subject-specific knowledge.

On the other hand, the difference in performance for the personalized labels gives an indication of how well the personal model adapts to both the introduction of a previously unseen subject and the adaption of the labels. In this situation, the difference is similar for P008, namely 0.013, but notably bigger for P041, namely 0.309.

	General Model		Personal Model	
	Unpersonalized	Personalized	Unpersonalized	Personalized
P008	0.655 ± 0.079	0.531 ± 0.027	0.688 ± 0.042	0.544 ± 0.157
P041	0.750 ± 0.031	0.611 ± 0.034	0.686 ± 0.027	0.302 ± 0.069

Table 8: Mean accuracy and standard deviation of the general and personal model trained on 140 samples, with a distinction between unpersonalized and personalized data.[35]

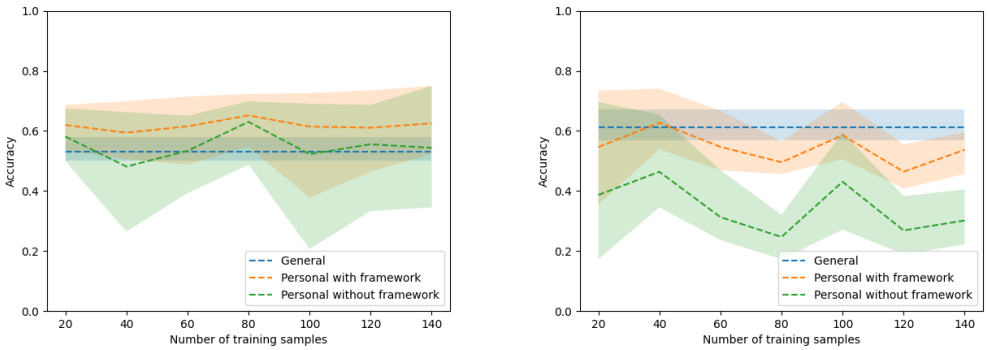
Now that the performance of the personal and general models has been presented, the next section will show the results with the inclusion of the framework. To get better insights

[35]Please note that the general model is always trained on unpersonalized data, so the personalization of the data only refers to the test data. The personal model is both trained and tested on either the unpersonalized or personalized data.

into the effect of the number of feedback samples received a sample size analysis plot was made.

Figure 22 shows that the performance is slightly increased for P008, although the effect is minimal. Since the difference is relatively small, and the number of cross-validation folds is low, the difference would most likely not be significant. For P041, the inclusion of the framework yields a bigger increase in accuracy. The reason why the framework adds more value in P041 than in P008 will be explained in Section 7.1.1.

Ideally, the framework would always yield performance which is equal to, or better than, the best model because it would choose the best-performing model for each sample. However, Figure 22b shows that the model with the framework often performs worse than the general model for P041.



(a) P008 (b) P041

Figure 22: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) of the personal model without the framework, the personal model with the framework and the general model on 5 cross-validation sets for the (a) P008 and (b) P041.

To get a better insight into where the framework yields improvements upon the personal model, a prediction comparison table was made for each subject with $|T| = 140$. The results of this experiment are shown in Tables 9a and 9b.[36]

The framework yields few improvements over any class for P008. The biggest change is that 28.3% of the samples for *put on headphones* were corrected by the use of the framework. The performance in the other classes is similar regardless of the framework. For P041, more improvements can be seen, as *cutting nails*, *yawn*, *put on headphones* and *other* all yield an improvement of at least 20% of the samples when adding the framework.

[36]The numbers are expressed in percentages in the tables for clarity, but the raw data can be found in Table 18 in Appendix D.2.

		Personal model									
		Yawn		Cutting paper		Cutting nails		Put on headphones		Other	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
PM+F	Correct	0.569	0.088	0.679	0.015	0.300	0.117	0.567	0.283	0.283	0.117
	Incorrect	0.039	0.304	0.015	0.291	0.000	0.583	0.000	0.150	0.033	0.567

(a) P008

		Personal model									
		Yawn		Cutting paper		Cutting nails		Put on headphones		Other	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
PM+F	Correct	0.218	0.208	0.414	0.138	0.200	0.400	0.333	0.383	0.197	0.262
	Incorrect	0.000	0.574	0.043	0.405	0.017	0.383	0.000	0.283	0.000	0.541

(b) P041

Table 9: Portion of the correctly and incorrectly predicted samples from the personal model and personal model with the framework (PM+F) in percentages for (a) P008 and (b) P041.[37]

Similarly to Section 6.1.2 and 6.1.3 on the toy problem, the following experiments on LF and AL will only report the performance with the framework included.[38]

6.2.2 Limited Feedback

Following the same approach as Section 6.1.2 on the toy problem, five LF techniques (*Correct*, *Sampled*, *Modified*, *URE* and *DNPL*) were compared to a *Full-Feedback* model. While different values for e will be tested later, the first experiment uses only binary feedback, i.e. $e = 1$.[39] To get insights into the training process, Figure 23 shows a sample size analysis plot. The *Full-Feedback* model yields only a small increase in accuracy in comparison to the other LF techniques. Furthermore, for P008, *Sampled* and *Correct* yield a slightly lower accuracy for $20 < |T| < 140$. However, the difference from *Sampled* and *Correct* to the other techniques is quite small. It is likely that this difference would not be significant if tested. *Sampled*, *Modified* and *DNPL* all perform similarly for P008. Additionally, for P041, all LF techniques perform similarly to each other.

As the training set size increases, the accuracy stays relatively constant for all LF techniques for P008. For P041, on the other hand, the accuracy decreases for all LF techniques as $|T|$ increases until 80 samples, after which it increases again.[40] The reason for the relatively unchanged performance, regardless of $|T|$, will be discussed in Section 7.1.1.[41]

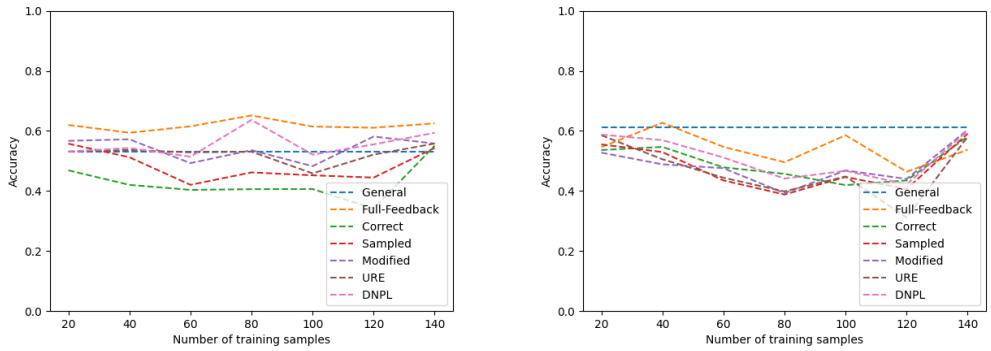
[37] *Sniff* and *staple book* are excluded because there are no samples from these classes.

[38] While the experiments were also conducted without the framework, only the results with the framework will be denoted in the following sections for clarity. The results of these experiments without the framework and a precision-recall plot can be found in Appendix D.1.

[39] Further explanation on the meaning of e can be found in Section 5.1.2.

[40] The relatively large jump between $|T| = 120$ and $|T| = 140$ caused by the framework, as the increase is not seen in the performance of the techniques without the framework in Appendix D.1.

[41] The variation in performance is similar for all techniques and can be found in Appendix D.3.



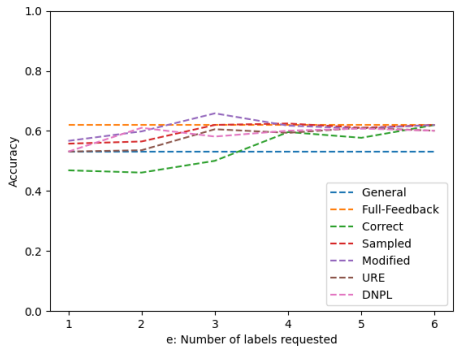
(a) P008 (b) P041

Figure 23: Mean accuracy of the LF techniques *Correct*, *Sampled*, *Modified*, *URE* and *DNPL* in comparison to the *Full-Feedback* model and general model with $e = 1$ for (a) P008 and (b) P041.

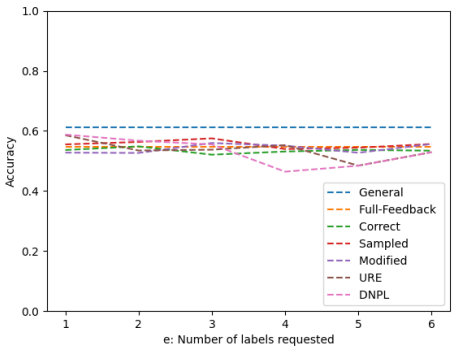
To get better insights into how the performance changes of each LF technique when the number of classes increases, a label request analysis plot was made with e ranging from 1 to 6. This helps to find the optimal value of e and to compare the LF techniques in different situations. Additionally, to get a better insight into how the performance for a value e is affected by the number of samples, this experiment was performed for both $|T| = 20$ and $|T| = 140$.

Figure 24 shows the results of this evaluation. With $|T| = 20$, most techniques perform similarly regardless of the subject. Interestingly, they also perform similarly to the *Full-Feedback* model, while the *Full-Feedback* always receives hard labels and the LF techniques receive limited feedback. Only the *Correct* technique yields a lower accuracy for P008 than the general model when $e < 4$. Remarkably, apart from the *Correct* technique for P008, the techniques perform similarly regardless of e.

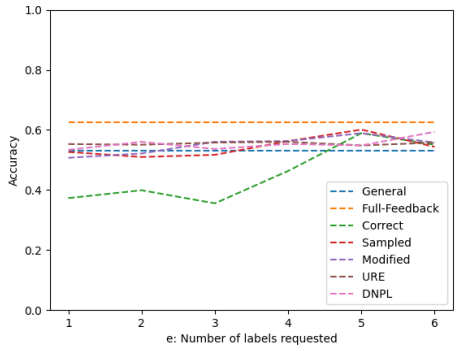
With $|T| = 140$, the difference between the *Full-Feedback* model and the LF techniques increases for both subjects. While the accuracy of the *Full-Feedback* model is relatively unaffected by $|T|$, the accuracy of the LF techniques decreases for an increased $|T|$. The reason for this decreased performance will be discussed in Section 7.1.2. Additionally, the *Correct* technique seems to decrease more than the other techniques for P008, although it is uncertain whether this change is due to variance in performance, or whether this change would generalize. A lower performance for the *Correct* technique would not be surprising, as will be explained in Section 7.1.2.



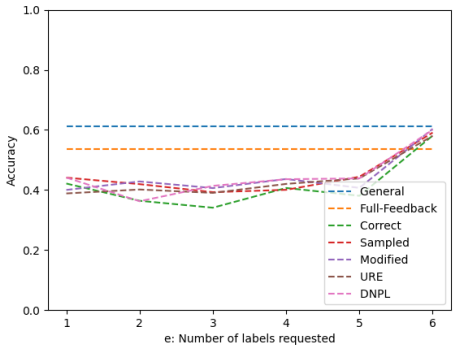
(a) $|T| = 20$, P008



(b) $|T| = 20$, P041



(c) $|T| = 140$, P008



(d) $|T| = 140$, P041

Figure 24: Mean accuracy of the LF techniques *Correct*, *Sampled*, *Modified*, *URE* and *DNPL* in comparison to the *Full-Feedback* model and general model with e varied between 1 and 6 with five-fold cross-validation, with $|T| = 20$ and $|T| = 140$ for P008 and P041.

To get an insight into how well the techniques perform for each class, Table 10 shows a class-specific performance table with $|T| = 140$ and $e = 1$. For P008, the LF techniques yield the highest F1-scores in the classes where the *Full-Feedback* model also yields the highest F1-scores, namely *yawn*, *cutting paper* and *put on headphones*. When excluding the *Full-Feedback* model, the highest F1-scores are yielded by *DNPL* for *cutting paper*, *cutting nails* and *other* and by *Modified* for *yawn* and *put on headphones*. However, the F1-scores are often very close, e.g. *cutting paper* yields an F1-score of 0.570 for *DNPL* and 0.561 for *Sampled*. Since the F1-scores are often very close to each other, it is likely that the higher performance of *DNPL* and *Modified* would not be significant if tested.

	Full-Feedback			*Correct*			*Sampled*			*Modified*			*URE*			*DNPL*		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Yawn	0.691	0.657	0.673	0.680	0.333	0.447	0.629	0.382	0.476	0.595	0.735	**0.658**	0.567	0.500	0.531	0.708	0.500	0.586
Cutting paper	0.612	0.694	0.650	0.489	0.478	0.483	0.503	0.634	0.561	0.564	0.493	0.526	0.535	0.507	0.521	0.529	0.619	**0.570**
Cutting nails	0.610	0.417	0.495	0.279	0.200	0.233	0.371	0.217	0.274	0.320	0.267	0.291	0.321	0.150	0.205	0.462	0.300	**0.364**
Put on headphones	0.718	0.850	0.779	0.737	0.467	0.571	0.516	0.550	0.532	0.826	0.633	**0.717**	0.486	0.583	0.530	0.677	0.700	0.689
Other	0.533	0.400	0.457	0.288	0.500	0.366	0.537	0.367	0.436	0.538	0.350	0.424	0.345	0.317	0.330	0.491	0.450	**0.470**

(a) P008

	Full-Feedback			*Correct*			*Sampled*			*Modified*			*URE*			*DNPL*		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Yawn	0.642	0.426	0.512	0.574	0.347	0.432	0.565	0.257	0.354	0.618	0.416	**0.497**	0.857	0.238	0.372	0.559	0.327	0.413
Cutting paper	0.478	0.552	0.512	0.495	0.457	0.475	0.465	0.397	0.428	0.445	0.422	0.434	0.348	0.345	0.346	0.395	0.603	**0.478**
Cutting nails	0.632	0.600	0.615	0.636	0.350	0.452	0.697	0.383	0.495	0.562	0.300	0.391	0.842	0.533	**0.653**	0.538	0.350	0.424
Put on headphones	0.558	0.717	0.628	0.479	0.567	0.519	0.485	0.550	0.516	0.353	0.400	0.375	0.871	0.450	**0.593**	0.765	0.217	0.338
Other	0.528	0.459	0.491	0.471	0.541	**0.504**	0.320	0.393	0.353	0.508	0.492	0.500	0.312	0.656	0.423	0.441	0.426	0.433

(b) P041

Table 10: Precision (Prec), recall (Rec) and F1-score (F1) in percentages for each class for the LF techniques *Correct*, *Sampled*, *Modified*, *URE* and *DNPL* in comparison to the *Full-Feedback* model with $e = 1$ for (a) P008 and (b) P041. The highest F1-score per class, excluding the *Full-Feedback* model, is highlighted in bold.[42]

[42] *Sniff* and *staple book* were not included, because there are no samples from these classes.

Whereas the LF techniques follow a similar trend to the *Full-Feedback* model for P008, this does not occur for P041. Each technique has its own class it performs best in, e.g. *cutting nails* and *put on headphones* for the *Full-Feedback* model, but *cutting paper* for *DNPL*. This also leads to the highest F1-scores being more spread among techniques, because *Modified* yields the highest F1-score for *yawn*, *DNPL* for *cutting paper*, *URE* for *cutting nails* and *put on headphones* and *Correct* for *other*. However, similarly to P008, the scores are often very close, so it cannot be concluded that one technique outperforms another in these classes.

Apart from only evaluating based on the accuracy, it is interesting to see if the techniques differ in convergence rate. If multiple LF techniques yield a similar accuracy, but one yields this performance in fewer epochs, that technique might be preferable to use in this situation. Therefore, a convergence rate table was made with $e = 1$. Table 11 shows that the average number of epochs generally varies between 6 and 7. *DNPL* yields a slightly higher number of epochs for P008, and *URE* for P041, although the difference to other techniques is minor. Therefore, it cannot be concluded that some techniques converge quicker than others.

	Full-Feedback	*Correct*	*Sampled*	*Modified*	*URE*	*DNPL*
P008	6.4 ± 3.6	6.8 ± 4.0	6.4 ± 2.1	5.4 ± 4.6	5.8 ± 4.7	8.8 ± 3.9
P041	6.0 ± 3.0	6.0 ± 2.6	6.0 ± 3.0	6.6 ± 2.7	8.6 ± 3.7	5.2 ± 2.8

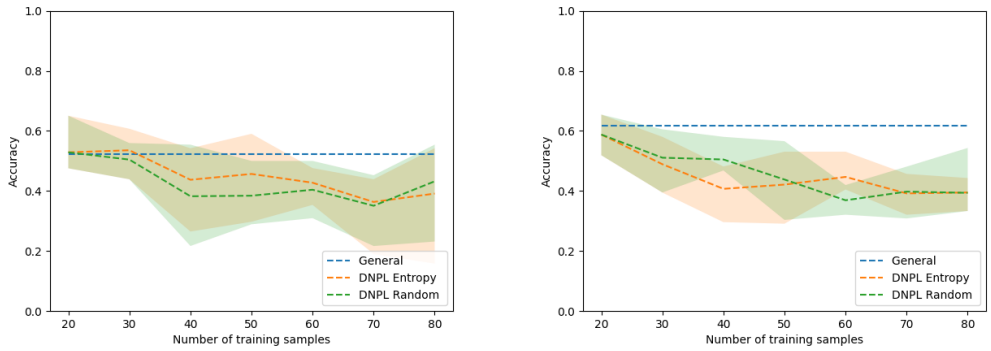
Table 11: Mean number of epochs required to yield the lowest validation loss, along with the standard deviation, of the LF techniques on P008 and P041.

6.2.3 Active Learning

Following the same approach taken for the toy problem in Section 6.1.3, two AL strategies were compared, namely random sampling and entropy-based sampling. As described in Section 5.1.3, the experiments were performed with the best-performing LF technique. Because it could not be concluded that one LF technique outperforms the others, a technique was chosen that performed relatively well for both subjects, namely *DNPL*.

To get an insight into the training process, Figure 25 shows a sample size analysis plot. For both P008 and P041, the two strategies follow the same trends with minor differences between the two strategies. For P008, the performance decreases as the number of samples increases for both strategies, after which the performance increases again at $|T| = 80$. For P041, the performance decreases steadily as the number of samples increases. These observations are similar to the results of most LF strategies between $|T| = 20$ and $|T| = 80$, as was previously shown in Figure 23.

An additional goal was to establish the number of samples to yield an optimal performance. Both strategies cannot improve upon the general model within $|T| = 80$ for both subjects, making an optimal performance difficult to establish. The reason for not reaching an optimal performance will be discussed in Section 7.1.1. When keeping these parameters, and a maximum of $|T| = 80$, the highest accuracy is reached by minimizing $|T|$ at 20 samples.



(a) P008 (b) P041

Figure 25: Mean accuracy of entropy-based and random sampling with *DNPL* with $e = 1$ for (a) P008 and (b) P041.

To illustrate how well each sampling strategy performed in terms of classes, Table 12 shows a class-specific performance table with $|T| = 80$. For P008, random sampling yields a higher F1-score for *yawn*, *cutting paper* and *other*, while entropy-based sampling yields a higher F1-score for *cutting nails* and *put on headphones*. Interestingly, this division is entirely swapped for P041, where random sampling yields a higher F1-score for *cutting nails* and *put on headphones* and entropy-based sampling for the remaining classes. It is important to mention that the scores are relatively close for most classes, and therefore it is not possible to say that one strategy outperforms the other in a class. In general, the difference between random and entropy-based sampling is minimal.

This concludes this chapter since the results of the framework, LF and AL have all been presented on both the toy problem and HAR dataset. The next section will discuss these results, compare the results with the literature and identify limitations.

	Random			*Entropy*		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Yawn	0.633	0.304	**0.411**	0.533	0.314	0.395
Cutting paper	0.527	0.515	**0.521**	0.475	0.425	0.449
Cutting nails	0.290	0.150	0.198	0.400	0.317	**0.355**
Put on headphones	0.524	0.733	0.611	0.868	0.550	**0.673**
Other	0.551	0.450	**0.495**	0.338	0.367	0.352

(a) P008

	Random			*Entropy*		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Yawn	0.431	0.307	0.358	0.436	0.436	**0.436**
Cutting paper	0.469	0.328	0.386	0.440	0.379	**0.407**
Cutting nails	0.358	0.633	**0.458**	0.397	0.383	0.390
Put on headphones	0.605	0.383	**0.469**	0.559	0.317	0.404
Other	0.466	0.443	0.454	0.574	0.443	**0.500**

(b) P041

Table 12: Precision and recall in percentages for each class for random and entropy-based sampling for (a) P008 and (b) P041, with the highest F1-score in bold.[43]

[43] *Sniff* and *staple book* were not included, because there are no samples from these classes.

7 Discussion

This study has compared methods for the effective use of LF (limited feedback) on a toy problem and a HAR (Human Activity Recognition) dataset. The results of the experiments have been presented in the previous chapter, whereas this chapter will highlight and discuss the most interesting results. This consists of possible explanations for unexpected results and possible reasons for the difference in performance between techniques. Furthermore, the results will be placed in context with related work, which was discussed in Chapter 3.

Similarly to the previous chapter, the framework, LF, and AL (Active Learning) will be discussed separately. However, instead of discussing these topics separately for the toy problem and HAR, each topic discusses both datasets. Because the discussion of the framework's performance is affected by the performance of the personal and general models, the section on the framework splits up the results between the toy problem and HAR. The sections on LF and AL, on the other hand, discuss the results from the toy problem and HAR together to allow comparisons between the datasets. Importantly, multiple factors are different between the two datasets, such as the dimensionality, number of classes and model size. Therefore, an observed difference could be attributed to any of these factors, but can nevertheless lead to interesting insights. Apart from discussing the results seen in the previous chapter, also the limitations of this study will be discussed.

7.1 Discussion of the Results

7.1.1 Framework

As mentioned previously, the results of the toy problem and HAR will be discussed separately for the framework. Within each dataset, the performance of the personal and general models will be discussed, along with the capabilities and limitations of the ND (Novelty Detection) and DCS (Dynamic Classifier Selection) components. Lastly, the performance of the framework will be discussed.

Toy Problem There are no major topics of discussion considering the personal and general models for the toy problem.[44] Therefore, this section continues with a discussion on the functioning of the ND and DCS components. Figure 14 in Section 6.1.1 has shown that ND can select novel samples relatively well for the toy problem, but also that the functioning

[44] One minor finding is that the personal model performs better for the drift and combined datasets than for the unexplored dataset. This difference can, most likely, be attributed to the required change in the model parameters. The change for the drift dataset is relatively minor since the decision boundaries of the general model only need to be shifted slightly. Additionally, for the combined dataset, one output possibility never occurs, meaning that the model can learn to ignore one of the classes, making the remaining choice easier. The unexplored dataset is a bit more complex in comparison because it still involves three output possibilities, but also a relatively large change in the decision boundaries. This is most likely the reason why the personal model requires more data to learn the distribution of the unexplored dataset than for the drift and combined datasets.

of a distance measure may be dependent on the dataset. For example, the toy problem is a 2D space, which is not always perfectly separable by cosine distance. Because of this, some samples are selected as non-novel, while they are not similar in terms of features, and resemble another class. This shows that careful selection of the distance measure is essential to the functioning of the framework.

In terms of DCS, Figure 14 in Section 6.1.1 has shown that, for the toy problem, the personal model was chosen for almost all samples for the unexplored dataset. This makes sense because the personal model improves over the general model in all three datasets. Additionally, if the personal model and general model both yield the same accuracy in the region of competence, the personal model is chosen. Therefore, in regions where the models perform similarly, the personal model is selected as a form of tiebreaker rather than due to its high performance.

Since the toy problem consists of low-dimensional datasets with a small model, the expectation was that the use of the framework would yield a large performance increase. However, Figure 15 has shown that the addition of the framework yields a similar performance as the personal model on all three datasets.[45] In retrospect, it makes sense that there was no performance increase or only a small increase. If the personal model simply improves upon the general model, the best choice for the framework is to always choose the personal model. Therefore, the maximal performance that can be reached is equal to the performance of the personal model.[46]

HAR While it was expected that the personal model would improve upon the general model, the personal model's performance is similar to the general model for subject P008 and lower for subject P041, as was shown in Section 6.2.1. This is most likely caused by a phenomenon called catastrophic forgetting, where neural networks forget previously learned information. An example of this is that the personal models often confused *yawn* with *cutting paper*, while the general models did not suffer from this problem. Since both the train and test accuracy are low, this is most likely not caused by overfitting. Instead, the learning rate may have been too high, causing the model to oscillate around the minimum or overshoot it,

[45]The use of the framework has also been evaluated per class, which showed that it can be more useful for certain classes than for others. Specifically, Table 4c has shown that for the unexplored dataset, the framework is unable to improve upon the *breakfast* and *dinner* classes but can correct 22.1% of the predictions for the *lunch* class. Since the *dinner* class was changed in the unexplored dataset, it might seem counter-intuitive that the framework cannot improve the performance for that class. However, since the general model is unable to classify the *dinner* class well, the best choice for the framework is to always comply with the personal model's predictions, and thus, does not improve upon the personal model.

[46]The framework has yielded lower performance than the personal model in specific situations. Figure 15a has shown that when training on a few samples, the personal model with framework yields lower performance than the personal model without framework on the drift dataset. When the number of training samples is low, most samples are classified as novel, and thus classified by the general model. Therefore, if the personal model outperforms the general model after just a few samples, the framework will be outperformed by the personal model. While this might indicate that the framework is not beneficial to use for few samples, the advantage of this method is that it decreases the performance variance when training with few samples.

rather than converge towards it. Therefore, decreasing the learning rate could have improved the performance [135]. Another change that could have improved the performance is to include warm-up in the training process, which can help to decrease drastic changes to the model when fine-tuning on new data [136]. In general, extensive hyper-parameter tuning would, most likely, have improved the performance.

Section 6.2.1 showed a difference in performance for the personal models between P008 and P041, where the personal model for P008 performed better.[47] Figure 20 shows that the personal model for P041 predicts many samples as *cutting paper* incorrectly. This is most likely caused by the fact that most samples from *staple book* were relabelled as *cutting paper*, therefore making the *cutting paper* class approximately twice as big as the other classes. This led to an imbalance in the dataset, which was not accounted for in the model. Imbalance methods, such as re-sampling or cost-sensitive learning, could have improved the performance overall [137]. It is uncertain why the personal model for P041 suffers more from this than for P008, but it could be caused by the fact that the training set for P041 was slightly smaller than for P008, or possibly because P041 moves more differently from the general population than P008.[48]

Importantly, because the performance for the HAR dataset is relatively low overall, this results in similar performance across the LF techniques and AL strategies. Additionally, some methods may suffer less from catastrophic forgetting, which does not necessarily mean they are the best method to use in a scenario where the models do not suffer from catastrophic forgetting. Therefore, it is important to consider that the results may not generalize towards other situations, models and datasets.

Now that the performance of the general and personal models has been discussed, the following paragraphs focus on the performance when including the framework. As opposed to the toy problem, the addition of the framework improves the performance for both P008 and P041 (see Figure 22 in Section 6.2.1). As explained previously, since the personal model already performs well for all classes in the toy problem, the chance is smaller that the framework adds value to the personal model. For HAR, however, the personal model performs worse than the general model, meaning that the framework can improve performance by choosing the general model for part of the samples. This also shows that the framework is especially useful if the personal model forgets knowledge on at least one of the classes, which could be seen for the unexplored and HAR datasets.

[47] As shown in Section 6.2.1, there are also minor differences in performance for the general model between P008 and P041, where the general model performs worse for P008. The class it performs worst at is *cutting nails*. Even though this is not a personalized class, the poor predictions of the model for *cutting nails* make sense, because *cutting nails* is very similar to *cutting paper* and *staple book*, which increases the difficulty of the class.

[48] Section 6.2.1 also showed that there is a bigger difference in performance from unpersonalized to personalized data for P041 than for P008. This shows that, for P041, label adaption posed a greater difficulty than adapting to an unseen subject. This makes sense because the data was acquired in a controlled environment, so there is a high probability that all subjects performed the activities relatively similarly, which can lead to a smaller effect of subject-specific knowledge.

The improvements from the framework are relatively evenly spread over the classes, where the framework corrects more samples for P041 than for P008, as could be seen in Table 9. This makes sense because the personal model for P041 performs worse than for P008, meaning that the general model is more likely to perform better than P041. Interestingly, this means that even for the personalized classes (*yawn* and *cutting paper*), the framework can improve upon its performance. However, a large proportion of samples is still classified incorrectly from these classes, which most likely include most of the samples with personalized labels.

While the framework performs similarly to or better than the personal model for every dataset, it does not always perform similarly to the general model. Figure 22b has shown that the framework can yield lower performance than the general model, even though it is expected that the framework performs at least as well as the best-performing model. This may be because the personal model is chosen if the personal model and general model yield the same performance in the region of competence. Since the region of competence uses $k = 2$, the chance of yielding the same number of correct predictions is high, therefore the chance of selecting the personal model is high, even though it is more likely to perform worse than the general model. While a k-NN (k-Nearest Neighbours) has shown satisfactory performance in ND [71], [75], [76] and DCS [58], other methods for ND and DCS might perform better and would yield a performance at least equal to the best-performing model.

In general, the framework can improve upon the personal model when the personal model lacks performance. This is clearly an advantage, specifically in a healthcare setting, where a lacking model could affect a patient's well-being. This improvement is shown for two very diverse datasets, i.e. a toy problem and HAR dataset, which shows promise for its applicability to other problems, rather than being limited to function on one dataset only. Apart from that, it is important to acknowledge that the framework is a novel approach. This framework could possibly be used in more studies, especially in fields where it is essential that a personal model should not lack performance. While it has shown promise in this study, more research is needed to optimize the framework and test it in different situations. The next sections will discuss the results on LF and AL, which both used the framework in their results.

7.1.2 Limited Feedback

This section provides a discussion on the results of the LF techniques on the toy problem (Section 6.1.2) and HAR (Section 6.2.2). The LF techniques consist of *Correct*, *Sampled*, *Modified*, *URE* and *DNPL*, which were all compared to a reference model which received hard labels for each sample, i.e. the *Full-Feedback* model. This section will first discuss general remarks concerning all techniques. Afterwards, the *Full-Feedback* model will be discussed in comparison to the LF techniques. Moreover, the influence of e, i.e. the number of labels requested, will be discussed. Lastly, each LF technique will be discussed separately and compared to the state-of-the-art.

First, it is important to mention that all LF techniques yield a relatively low performance for the HAR dataset. This is, most likely, due to the same problem addressed in the previous section; catastrophic forgetting. Importantly, this also means that the performance of the techniques might be more influenced by their ability to counter catastrophic forgetting than by how well the technique is able to learn new information.

Second, the framework combines knowledge from both models. Therefore, if the models perform poorly, the influence from the general model increases, which diminishes the differences in performance. For more insights, a study should be conducted to investigate how the techniques perform without the framework.

Regardless of the dataset, the *Full-Feedback* model performs similarly to the other LF techniques, as can be seen in Figures 16 and 23. Remarkably, this suggests that the LF techniques can train just as well from soft as hard labels. However, it is uncertain if this observation would generalize towards new datasets and hyperparameters. For the toy problem, it may be easier for the LF techniques to yield high performance with soft labels because of the small number of classes. With 3 classes, the chance of receiving a hard label is high, and if a soft label is retrieved, there are only two remaining possibilities, which can make learning from a soft label easier. Next to that, for HAR, all techniques suffer from catastrophic forgetting, which can limit the ability of the *Full-Feedback* technique to significantly outperform the LF techniques. Therefore, it could be said that they can all handle catastrophic forgetting similarly, as opposed to being able to train similarly well. This is also a possible explanation for why the LF techniques do generally not benefit from a higher e (see Figure 24), i.e. a higher number of requested labels. These explanations indicate that it may be possible that there would be a greater difference between the LF techniques and the *Full-Feedback* model.

However, it is also possible that the LF techniques can simply learn similarly well from LF as they can from hard labels. Lucas et al. [79] also found that the LF techniques they tested (*Correct*, *Sampled*, *Modified* and *DNPL*[49]) sometimes even outperformed the *Full-Feedback* model. Unfortunately, *DNPL* as proposed by Seo and Huh [96], and *URE* were not compared to a *Full-Feedback* model in their original studies. Other studies on PLL and CLL [81], [98] have shown that a *Full-Feedback* model performs better than their PLL and CLL techniques, especially when dealing with an increased number of (candidate labels in the) partial labels. This suggests that the results of this study may not apply to other datasets. On the other hand, the techniques tested in this study are not the same as those mentioned in [81] and [98]. Additionally, *DNPL* has outperformed one of the techniques mentioned in those papers in three out of four datasets [96]. Therefore, it is also possible that the other techniques simply perform worse.

The *Correct* technique yields lower performance than the other techniques in the toy problem (see Figure 16), and slightly lower performance in HAR (see Figure 23). Lucas et al. [79] also found that *Correct* yields low performance for a small model on a low-

[49]Referred to as *Uniform* in their study.

dimensional dataset, although they found that it does outperform the other techniques for a complex model on a high-dimensional dataset. The lower performance in this study for the high-dimensional dataset could be caused by the fact that *Correct* reduces the number of training samples, which could increase the effect of catastrophic forgetting. Additionally, the low performance on the low-dimensional dataset is most likely because *Correct* is not able to improve if it does not correctly classify any personalized samples. The low-dimensional toy problem has relatively clear, predefined boundaries, making it very unlikely that the general model predicts samples from a personalized class correctly. This results in *Correct* not gaining any new information on the personalized classes. Sometimes it might predict samples near the decision boundary correctly, causing it to shift in the right direction slightly. However, as could be seen from the inability to learn about the unexplored dataset, this is not likely to occur if the drift in data distribution is bigger.

Figure 24 has also shown that the *Correct* technique benefits from having a higher e for the HAR dataset for P008. For a higher e, more labels are presented to an annotator, therefore the chance increases of a correct guess, which leads to an additional training sample. In turn, a higher number of training samples can lead to a better performance.

The *Sampled* technique performs similarly to the *Full-Feedback* model on the toy problem (see Figure 16) and HAR (see Figure 23) datasets, which corresponds to the results found by Lucas et al. [79]. They also studied the *Modified*, *Correct* and *DNPL* techniques and found that *DNPL* outperformed *Sampled* for smaller models on four low-dimensional datasets and that *Correct* outperformed *Sampled* for complex models on two high-dimensional datasets. These differences were not found in this study, where the performance of the techniques was similar to each other. This may be caused by the fact that the models were pretrained in this study, which reduces the effect of the difference in performance because all models are able to learn quickly. In contrast, the personal model yielded an overall low performance for the personal model for the HAR dataset, therefore also reducing the difference in performance between the models.

The *Modified* technique performs similarly to the *Full-Feedback* model on both the toy problem (see Figure 16) and HAR dataset (see Figure 23). Lucas et al. [79] found the same results for a high-dimensional dataset but found that *Modified* performs poorly on a smaller model on a low-dimensional dataset. Unfortunately, they have not reported a hypothesis for why *Modified* performs poorly on their small model with a low-dimensional dataset. This difference in results could be caused by the difference in setup between the experiments. The experiment by Lucas et al. did not focus on personalization, so instead, they used a fully labelled pretraining batch of 20 samples, after which the feedback phase was started. This means that the goal of the models is to learn the general distribution of the data, whereas in this study the goal of the models is to adapt to changes in the data distribution. It could be that the *Modified* technique is not well fitted to learn a new distribution, but can be used to adapt itself gradually. Another reason for the difference in results could be the number of classes. Lucas et al. tested with five, ten and fifteen different classes, where, as the number of

clusters increased, *Modified*'s performance decreased. In this study, only three classes were used, therefore higher performance may be seen.

The *URE* performs similarly to the *Full-Feedback* model on the toy problem (see Figure 16) and HAR (see Figure 23). Previous studies have not compared the *URE* to any other LF techniques used in this study, making it more difficult to compare the results to the literature. Feng et al. [99] presented the *URE* and showed that it performs well for complex models for five high-dimensional datasets. This study finds similar results for a complex model for a high-dimensional dataset and adds that the *URE* also shows promise for a smaller model for a low-dimensional dataset.

DNPL performs similarly to the *Full-Feedback* model on both the toy problem (see Figure 16) and HAR dataset (see Figure 23). Lucas et al. [79] presented *DNPL*[50] and found similar results, in the sense that it performed well on both a smaller model for low-dimensional datasets and a more complex model for a high-dimensional dataset. One difference in observations is that they report higher performance from the *Correct* technique than *DNPL* for a complex model on a high-dimensional dataset, whereas this study reports similar performance between the *Correct* technique and *DNPL*. Again, this might be caused by the fact that the differences in performance are decreased due to all techniques suffering from catastrophic forgetting. Apart from Lucas et al., Seo and Huh [96] also proposed *DNPL*, but in the context of PLL (Partial Label Learning). Within PLL, generally, a small number of candidate labels is used, i.e. 2 to 3. This study suggests that *DNPL* can also perform well when using 4 to 5 candidate labels.

Importantly, these LF techniques have never all been compared to each other in previous studies. While Lucas et al. [79] have compared multiple techniques, they have not considered a comparison to PLL and CLL (Complementary Label Learning) techniques, and instead, only compare their own techniques. Similarly, PLL and CLL techniques are generally only compared to techniques from their own field of study. This study compares techniques using a probability distribution to techniques from the field of PLL and CLL, which has not been done before.

The next section discusses the results regarding AL. Notably, since AL used an LF technique, the discussed points above are also relevant to the following section.

7.1.3 Active Learning

In both the toy problem and HAR dataset, minor differences are seen between entropy-based and random sampling (see Figure 19 and 25). Other studies that have compared the two methods generally report an outperformance of entropy-based sampling over random sampling, although similar performances are also reported occasionally [52], [138]. It is important to consider that "the winning strategy is different under different circumstances" [104, p. 937]. This indicates that it is possible that the circumstances of this study, i.e. using an LF technique on a personalized dataset, cause random sampling to perform similarly to

[50]The technique was referred to as *Uniform* in their study.

entropy-based sampling. The following paragraphs will describe some characteristics that can influence the performance of the strategies as opposed to other studies.

The first characteristic is that the personal model inherits the weights of the general model. Entropy-based sampling may select samples which were originally the most uncertain for the general model, but not for the personalized data. For example, entropy-based sampling may be more likely to select samples near a decision boundary, while, in this case, it may be more useful to sample from a personalized class. Random sampling may be more likely to select samples from a personalized class and adapt quicker than entropy-based sampling. To counter this, AL strategies specifically aimed at handling changes in the data might be more effective [139], [140].

The two datasets also have their own characteristics that might influence the strategies' performance. The toy problem consists of only three classes, which results in a high chance of random sampling from a personalized class. In turn, this may lead to both strategies adapting after a similar number of samples. On the other hand, for HAR, both strategies most likely suffer from catastrophic forgetting, which could result in similar performances regardless of the selected samples.

Another characteristic of this study is the use of a LF technique. Selecting the most uncertain sample has a greater chance of leading to a soft label. In turn, a soft label may be less informative than a hard label. Therefore, it might be more beneficial to sample randomly and decrease the chance of a soft label, as opposed to entropy-based sampling.

As far as the author could ascertain, this is the first study that applies AL strategies to LF techniques. Initial findings suggest that entropy-based sampling cannot lead to a reduction in the number of feedback samples compared to random sampling. While more research is needed to see if the results generalize, this study provides first insights into the performance of random and entropy-based sampling with an LF technique in the context of personalization. While this study provides new insights, it also comes with limitations.

7.2 Limitations

One of the main limitations of this study is the small number of datasets and subjects used. Specifically, the evaluated techniques were only tested on one HAR dataset for two subjects, which is insufficient to conclude that the performances of the techniques generalize towards other subjects or datasets. Moreover, none of the results were verified using statistical analyses, therefore it is not possible to determine if the differences in performance are significantly different.

Next to that, as mentioned before, the poor performance of the models on the HAR dataset may have reduced the performance difference between the LF and AL techniques. Future studies should be conducted to find how the techniques perform when the overall performance is higher.

Regarding personalization, the experiments on the toy problem included three types of personalization: the combination of two classes, drift, and samples from an unexplored

location in the input space. While these types of personalizations simulate possible real-life situations, it is uncertain whether they occur in the real world. Additionally, only one type of personalization was tested for the HAR dataset, namely the combination of two classes. A different type of personalization, such as drift, might lead to different results. Therefore, the results cannot be generalized towards personalization as a whole, but only to this specific type of personalization.

Furthermore, each technique was tested with the underlying framework, which was not extensively tested in this study or related work. Therefore, it is not clear how the framework would perform in other datasets. Additionally, the use of the framework increases performance for poor-performing methods, therefore diminishing the difference in performance that could be seen without the framework.

8 Conclusion and Future Work

This study researched effective ways to use LF (limited feedback) for personalization. To do so, a new framework was proposed that aimed to optimize the use of both the general and personal model. With the use of this framework, multiple LF techniques were evaluated to minimize annotation effort while maximizing the learning gain. Next to that, two AL (Active Learning) strategies were compared to reduce the number of feedback samples required to learn optimally. The experiments have been performed on a low-dimensional toy problem and a high-dimensional HAR (Human Activity Recognition) dataset.

This chapter provides an overview of each of the aforementioned topics and summarizes the findings of the experiments. Firstly, the design and performance of the framework will be discussed. After that, the sub-questions will be answered, and their answers will lead to the answer to the main research question. Lastly, suggestions for future work will be provided.

8.1 Research Questions

This study employed a novel framework that aims to combine the knowledge from a personal and general model optimally. This framework consists of a ND (Novelty Detection) and a DCS (Dynamic Classifier Selection) component. While the framework may not always outperform the general model, it generally did perform equal to or better than the personal model. In the toy problem, the personal model performed well, leading to minimal improvements from the framework. On the other hand, the personal model did not perform well in the HAR dataset, which allowed the framework to improve upon the personal model using knowledge from the general model. While it improved the performance of the personal model, it did not always yield better performance than the general model. Despite this, its results are promising and different implementations of the ND and DCS components should be further explored in future work.

After establishing the framework, experiments were conducted to answer the first sub-question (SQ1), which describes the trade-off between the maximization of learning gain and the minimization of the level of feedback. To find the answer to this question, five LF techniques (*Sampled*, *Correct*, *Modified*, *URE* and *DNPL*) were compared for differing levels of detail by varying values of e. Importantly, these techniques come from different fields of study, which were not compared yet before. A comparison of the techniques can give an indication of their applicability to LF. Additionally, they have never been tested in a study regarding personalization with varying values of e.[51]

[51] In previous works, *Sampled*, *Modified* and *Correct* were only tested with $e = 1$ as they originated from an approach for handling binary feedback. Next to that, *DNPL* was proposed from the field of PLL (Partial Label Learning), where candidate labels are generally related to each other, which is not the case in this study. *URE* stems from CLL (Complementary Label Learning), which has only been compared to other CLL approaches and one other PLL approach.

This study has shown that all LF techniques except for *Correct* performed similarly to the *Full-Feedback*[52] model for the three toy problem datasets and two subjects in the HAR dataset. Specifically, *Correct* failed to converge in all five folds for the unexplored dataset, and some of the folds for the drift and combined datasets. On the other hand, *Sampled*, *Modified*, *URE* and *DNPL* performed similarly to the *Full-Feedback* model, despite receiving only binary feedback. In the toy problem, these techniques showed improvement with an increase in the number of training samples, with the performance maximizing at $|T| = 100$. For HAR, the techniques performed similarly to the *Full-Feedback* model, although the overall performance was similar to, or lower than, the general model. Therefore, the techniques could not improve on the general model for either subject.

All techniques, except *Correct*, performed similarly regardless of the value of e. This suggests that minimizing the level of detail to $e = 1$ does not lead to a loss of learning gain for the other four techniques, with the hyperparameters used. However, it is worth noting that the models were unable to improve upon the general model, most probably due to catastrophic forgetting. As a result, these findings may not be applicable when the techniques perform better. In general, these results might only apply to the specific situation of this study and it is unclear whether these findings would generalize towards other datasets, subjects or when using different hyperparameters. Future research is necessary to determine if the level of detail in feedback can be minimized in a different experimental setup.

The second sub-question (SQ2) focuses on determining the minimal number of feedback samples required to yield optimal performance through the effective use of feedback, for which two AL strategies were compared: random and entropy-based sampling. Notably, AL has not been applied in combination with LF in previous works. Both strategies yield their maximum results at 7 to 8 samples for the drift dataset and 6 samples for the combined dataset. In the unexplored dataset, their performance peaks at 4 training samples, after which it starts to decline for entropy-based sampling. Importantly, the strategies both yield variations in performance between the accuracy of the general model and nearly optimal accuracy, i.e. 97%. Due to this, no conclusions can be made on the exact number of samples to train optimally. For the HAR dataset, both strategies cannot improve upon the general model within 80 training samples. Instead, their highest performance is reached by minimizing the number of training samples to 20.

In general, entropy-based and random sampling performed similarly and did not yield a difference in the number of samples required to yield an optimal level of performance for both the toy problem datasets and HAR dataset. It was expected that entropy-based sampling could use LF more effectively than random sampling, but these results indicate that entropy-based sampling cannot reduce the number of samples required. Importantly, similarly to the first sub-question, the results might not generalize towards other datasets, subjects or hyperparameters used. On the other hand, the results suggest that entropy is not

[52]The *Full-Feedback* model is a model which receives hard labels for every sample, as opposed to the LF techniques.

the optimal method to use when handling LF. Therefore, the fact that entropy-based sampling could not outperform random sampling may provide a basis for future work to test more AL strategies.

As mentioned in Section 7.1.3, certain characteristics of this study may have caused entropy-based sampling to perform worse. For instance, entropy-based sampling selects the most uncertain samples, which also have a higher chance of sampling a soft label, which yields less information. To increase the chance of sampling a hard label, the chance of sampling a soft label could be taken into account along with the sample's entropy. By measuring the entropy of the top e classes instead of the uncertainty over all classes, the chance of sampling a hard label may increase. Furthermore, there may be many other AL strategies that could perform better than entropy-based sampling, which should be tested in future work.

With both sub-questions addressed, the main research question (RQ) can be answered. A framework has been proposed and evaluated in this study to combine the knowledge of a personal and general model. The framework consists of components that are already present in literature, but the design of the complete framework is novel. Although there is room for improvement, it has shown promising results and should be explored further in future work. The main advantage of the framework is that it can improve the performance of the personal model when it fails to perform well, which is particularly useful in a healthcare setting. A poorly performing system can further increase the workload of healthcare workers or decrease the well-being of patients.

Next to that, this study has shown that the *Full-Feedback* model and the LF techniques *Sampled*, *Modified*, *URE* and *DNPL* perform similarly in the three toy problem datasets and for two subjects in the HAR dataset, with the specific hyperparameters used. This suggests that LF could be used in certain situations to reduce annotator effort without sacrificing performance. Annotators, such as healthcare workers and patients, may not have the time or expertise to choose the correct label from a long list of classes. Therefore, reducing the number of options presented to an annotator could make a HAR system feasible.

Furthermore, despite the attempt to apply AL to use LF more effectively, entropy-based sampling did not prove to be more effective than random sampling in this study. As a result, the annotator effort could not be minimized with the use of AL.

Finally, by testing all methods on two very diverse datasets, the generalizability to other datasets increases. This is relevant because the experiments performed in this study are not only applicable to HAR but also to other fields concerning personalization, such as home systems and recommender systems.

8.2 Future Work

In general, the experiments in this study should be performed with more datasets and subjects to validate the results in different situations. However, this study also suggests multiple new areas to explore in future work, covering the framework, LF, AL and other directions.

Framework A framework was used to combine the qualities of the personal and general models. However, this framework has not been tested extensively in previous works. Additionally, while it has proven useful in this study, it did not always perform as desired.[53] Therefore, testing the framework with the same structure but different ND (Novelty Detection) and DCS (Dynamic Classifier Selection) components could increase its effectiveness.

Limited Feedback This study compared several LF techniques, but numerous CLL (Complementary Label Learning) and PLL (Partial Label Learning) methods were not tested.[54] These could be explored in future work to determine if any methods outperform those tested in this study.

Apart from that, this study used a fixed value for e (between 1 and 6) for each method when training the models. Future work should explore the possibility of varying e during the training phase to adjust it based on the model's certainty. When the model is certain about a prediction, e can most likely be decreased without losing information. On the other hand, a high e may be preferable if the model is uncertain about a prediction since it increases the chance of sampling a hard label.

Active Learning As mentioned above, this study tested only random and entropy-based sampling as AL strategies. While integrating AL into LF has the potential to further reduce the effort needed to personalize a model, this study did not find a reduction using entropy-based sampling. Future work should investigate different AL strategies to see if they can reduce the number of labelled samples.

Specifically, the characteristics of the framework and LF should be considered since they are likely to influence the performance of AL. As discussed previously, entropy-based sampling may lead to more soft labels. Therefore, the combination of entropy and the chance of sampling a soft label might increase performance. Apart from LF, the framework also influences the performance of AL and could be used to increase the performance. For example, the novelty of the samples can be taken into account to increase the diversity of the selected samples. Next to that, DCS could be used to select samples for which the model performs poorly on its neighbours.

Next to that, there are many more methods to use samples effectively than AL. For instance, methods that use unlabelled data should be explored in combination with LF, such as self-supervision [141] or label propagation of similar samples [142].

Other As mentioned in Section 1.3, this study cannot cover all potential directions into personalized learning from feedback. However, it encourages the exploration of these out-of-scope directions in future work. This includes addressing challenges such as dealing with

[53] The framework generally performed similarly to, or better than, the personal model, but was not always able to perform better than both models. See Section 7.1.1 for further explanation of the situations in which this occurred, and why this occurred.

[54] See Section 3.2 for more relevant CLL and PLL methods.

class imbalance, multi-label datasets, weak labels, real-time data processing, user interface design and handling newly emerging classes.

Future work should also explore the opportunities and challenges in labelling skeletal data. While the annotators for this study could perform the labelling task relatively easily, only two classes had to be labelled.[55] When the labelling tasks include more classes, especially fine-grained activities, it might not be feasible to label them adequately. This should be explored in future work.

Additionally, the annotators can have different levels of expertise. For example, healthcare workers might not have a lot of time, so when they do give feedback, it must be valued more, especially since they have more knowledge on the subject. On the other hand, the patients themselves or caregivers may be more likely to select the wrong label due to a lack of knowledge. The knowledge of annotators could be used to weigh the feedback, but also to decide on the question according to the expertise of the annotator [143]. This approach allows expert annotators to give feedback on difficult cases, while less-experienced annotators are only asked to give feedback on easier cases. This expertise rating should be explored in future work, in combination with LF and AL from this study.

Next to that, the task should be explored in different contexts, e.g. with multiple people or with more activities. Information on timing and location could also be used in the ML algorithm to improve the performance.

[55] See Section B for more information on the labelling task and observations during the task.

9 References

[1] "Uitkomsten werknemersenquête AZW, 4e kwartaal 2021," Centraal Bureau voor de Statistiek, Tech. Rep., Apr. 2022. [Online]. Available: https://www.cbs.nl/nl-nl/achtergrond/2022/17/azw-uitkomsten-werknemersenquete-4e-kwartaal-2021.

[2] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh, "Activity Recognition and Abnormality Detection with the Switching Hidden Semi-Markov Model," in *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, USA: IEEE, Jun. 2005. DOI: 10.1109/cvpr.2005.61.

[3] O. Aziz, E. J. Park, G. Mori, and S. N. Robinovitch, "Distinguishing the causes of falls in humans using an array of wearable tri-axial accelerometers," *Gait & Posture*, vol. 39, no. 1, pp. 506–512, Jan. 2014, ISSN: 0966-6362. DOI: 10.1016/J.GAITPOST.2013.08.034.

[4] W. e. S. Ministerie van Volksgezondheid, *Leefstijlmonitoring*. [Online]. Available: https://www.vilans.nl/kennisbank-digitale-zorg/technologieen/leefstijlmonitoring.

[5] A. D. Singh, S. S. Sandha, L. Garcia, and M. Srivastava, "Radhar: Human activity recognition from point clouds generated through a millimeter-wave radar," in *Proceedings of the Annual International Conference on Mobile Computing and Networking*, Los Cabos, Mexico: ACM, Oct. 2019, pp. 51–56, ISBN: 9781450369329. DOI: 10.1145/3349624.3356768.

[6] Y. Wang, H. Liu, K. Cui, A. Zhou, W. Li, and H. Ma, "M-Activity: Accurate and Real-Time Human Activity Recognition Via Millimeterwave Radar," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, vol. 2021-June, Toronto, Canada: IEEE, Jun. 2021, pp. 8298–8302. DOI: 10.1109/ICASSP39728.2021.9414686.

[7] E. Garcia-Ceja, M. Riegler, A. K. Kvernberg, and J. Torresen, "User-adaptive models for activity and emotion recognition using deep transfer learning and data augmentation," *User Modeling and User-Adapted Interaction*, vol. 30, pp. 365–393, Oct. 2020. DOI: 10.1007/S11257-019-09248-1.

[8] S. Stumpf, E. Sullivan, E. Fitzhenry, I. Oberst, W.-K. Wong, and M. Burnett, "Integrating rich user feedback into intelligent user interfaces," in *Proceedings of the International Conference on Intelligent User Interfaces*, Gran Canaria, Spain: ACM, Jan. 2008, pp. 50–59. DOI: 10.1145/1378773.1378781.

[9] B. J. Dietvorst, J. P. Simmons, and C. Massey, "Overcoming Algorithm Aversion: People Will Use Imperfect Algorithms If They Can (Even Slightly) Modify Them,"

Management Science, vol. 64, no. 3, pp. 1155–1170, Nov. 2018. DOI: 10.1287/mnsc.2016.2643.

[10] P. Siirtola and J. Röning, "Incremental Learning to Personalize Human Activity Recognition Models: The Importance of Human AI Collaboration," *Sensors*, vol. 19, no. 23, p. 5151, Nov. 2019. DOI: 10.3390/S19235151.

[11] B. Cvetković, B. Kaluža, M. Gams, and M. Luštrek, "Adapting activity recognition to a person with Multi-Classifier Adaptive Training," *Journal of Ambient Intelligence and Smart Environments*, vol. 7, no. 2, pp. 171–185, Jan. 2015. DOI: 10.3233/AIS-150308.

[12] T. Pietraszek, "Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection," in *Recent Advances in Intrusion Detection*, Sophia Antipolis, France: Springer, Sep. 2004, pp. 102–124. DOI: 10.1007/978-3-540-30143-1_6.

[13] N. Mairittha, T. Mairittha, and S. Inoue, "Improving activity data collection with on-device personalization using fine-tuning," *International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 255–260, Sep. 2020. DOI: 10.1145/3410530.3414370.

[14] D. Cohn, R. Caruana, and A. K. McCallum, "Semi-Supervised Clustering with User Feedback," in *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, S. Basu, I. Davidson, and K. Wagstaff, Eds., CBC Press, 2008, ch. 2, pp. 17–31, ISBN: 9781584889977. DOI: 10.1201/9781584889977.ch2.

[15] Y. Yang, E. Kandogan, Y. Li, P. Sen, and W. S. Lasecki, "A Study on Interaction in Human-In-The-Loop Machine Learning for Text Analytics," in *Joint Proceedings of the ACM IUI Workshops*, Los Angeles, USA: ACM, Mar. 2019.

[16] M. Cho, G. Lee, and S.-W. Hwang, "Explanatory and Actionable Debugging for Machine Learning: A TableQA Demonstration," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Paris, France: ACM, Jul. 2019, pp. 1333–1336. DOI: 10.1145/3331184.

[17] F. Gumus, C. O. Sakar, Z. Erdem, and O. Kursun, "Online Naive Bayes Classification for Network Intrusion Detection," in *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, Beijing, China: IEEE, Aug. 2014, pp. 670–674, ISBN: 9781479958771. DOI: 10.1109/asonam.2014.6921657.

[18] E. Garcia-Ceja and R. Brena, "Building personalized activity recognition models with scarce labeled data based on class similarities," *Lecture Notes in Computer Science*, vol. 9454, pp. 265–276, Dec. 2015. DOI: 10.1007/978-3-319-26401-1_25.

[19] J. Xu, L. Song, J. Y. Xu, G. J. Pottie, and M. Van Der Schaar, "Personalized Active Learning for Activity Classification Using Wireless Wearable Sensors," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 5, pp. 865–876, Aug. 2016. DOI: 10.1109/JSTSP.2016.2553648.

[20] J. H. Hong, J. Ramos, and A. K. Dey, "Toward Personalized Activity Recognition Systems with a Semipopulation Approach," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 101–112, Feb. 2016. DOI: 10.1109/THMS.2015.2489688.

[21] N. Sadri and G. V. Cormack, *Continuous Active Learning Using Pretrained Transformers*, Aug. 2022. DOI: 10.48550/arxiv.2208.06955.

[22] V. Mirchevska, B. Kaluža, M. Lustrek, and M. Gams, "Real-time Alarm Model Adaptation Based on User Feedback," in *19th European Conference on Artificial Intelligence*, J. Gama, P. Cortez, N. Marques, and M. F. Santos, Eds., Lisbon, Aug. 2010, pp. 39–43.

[23] R. San-Segundo, J. M. Montero, J. Moreno-Pimentel, and J. M. Pardo, "HMM Adaptation for Improving a Human Activity Recognition System," *Algorithms*, vol. 9, no. 60, Sep. 2016. DOI: 10.3390/a9030060.

[24] J. Parviainen, J. Bojja, J. Collin, J. Leppänen, and A. Eronen, "Adaptive Activity and Environment Recognition for Mobile Phones," *Sensors*, vol. 14, no. 11, pp. 20 753–20 778, Nov. 2014. DOI: 10.3390/S141120753.

[25] G. K.-M. Liu, "Perspectives on the Social Impacts of Reinforcement Learning with Human Feedback," *arXiv preprint*, Mar. 2023. DOI: 10.48550/arXiv.2303.02891.

[26] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, vol. 2017-December, Long Beach, CA, USA, Jul. 2017. DOI: 10.48550/arxiv.1706.03741.

[27] Z. Wang, X. Xiao, G. Warnell, and P. Stone, "APPLE: Adaptive planner parameter learning from evaluative feedback," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7744–7749, Oct. 2021. DOI: 10.1109/LRA.2021.3100940.

[28] A. B. Karami, A. Fleury, J. Boonaert, and S. Lecoeuche, "User in the Loop: Adaptive Smart Homes Exploiting User Feedback—State of the Art and Future Directions," *Information*, vol. 7, no. 2, p. 53, Jun. 2016. DOI: 10.3390/INFO7020035.

[29] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding," *Computing Research Repository*, 2019. DOI: 10.1109/tpami.2019.2916873.

[30] L. Xia, C. C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3D joints," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Providence, USA: IEEE, 2012, pp. 20–27, ISBN: 9781467316118. DOI: 10.1109/CVPRW.2012.6239233.

[31] P. Wei, Y. Zhao, N. Zheng, and S. C. Zhu, "Modeling 4D human-object interactions for event and object recognition," in *Proceedings of the International Conference on Computer Vision*, Sydney, Australia: IEEE, 2013, pp. 3272–3279, ISBN: 9781479928392. DOI: 10.1109/ICCV.2013.406.

[32] R. Chavarriaga, H. Sagha, A. Calatroni, *et al.*, "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, Nov. 2013, ISSN: 0167-8655. DOI: 10.1016/J.PATREC.2012.12.014.

[33] N. Daniel, F. Goldberg, and I. Klein, "Smartphone Location Recognition with Unknown Modes in Deep Feature Space," *Sensors*, vol. 21, no. 4807, Jul. 2021. DOI: 10.3390/s21144807.

[34] T. E. Boult, S. Cruz, A. R. Dhamija, M. Gunther, J. Henrydoss, and W. J. Scheirer, "Learning and the Unknown: Surveying Steps toward Open World Recognition," in *Proceedings of the 33rd Conference on Artificial Intelligence*, Honolulu, USA: AAAI, Jan. 2019, pp. 9801–9807. DOI: 10.1609/aaai.v33i01.33019801.

[35] C. Geng, S. J. Huang, and S. Chen, "Recent Advances in Open Set Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3614–3631, Oct. 2021, ISSN: 19393539. DOI: 10.1109/TPAMI.2020.2981604.

[36] R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Information Fusion*, vol. 41, pp. 195–216, May 2018. DOI: 10.1016/J.INFFUS.2017.09.010.

[37] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On Combining Classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, Mar. 1998. DOI: 10.1109/34.667881.

[38] J. Fierrez-Aguilar, D. Garcia-Romero, J. Ortega-Garcia, and J. Gonzalez-Rodriguez, "Adapted user-dependent multimodal biometric authentication exploiting general information," *Pattern Recognition Letters*, vol. 26, no. 16, pp. 2628–2639, Dec. 2005. DOI: 10.1016/J.PATREC.2005.06.008.

[39] J. Fierrez, A. Morales, R. Vera-Rodriguez, and D. Camacho, "Multiple classifiers in biometrics. part 1: Fundamentals and review," *Information Fusion*, vol. 44, pp. 57–64, Nov. 2018, ISSN: 15662535. DOI: 10.1016/J.INFFUS.2017.12.003.

[40] D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia, "Dynamic Selection of Classifiers in Bug Prediction: An Adaptive Method," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 3, pp. 202–212, Jun. 2017, ISSN: 2471285X. DOI: 10.1109/TETCI.2017.2699224.

[41] L. Xu, A. Krzyzak, and C. Y. Suen, "Associative switch for combining multiple classifiers," in *Proceedings of the International Joint Conference on Neural Networks*, Seattle, USA: IEEE, 1991, pp. 43–48, ISBN: 0780301641. DOI: 10.1109/IJCNN.1991.155146.

[42] G. Giacinto and F. Roli, "Dynamic classifier selection based on multiple classifier behaviour," *Pattern Recognition*, vol. 34, no. 9, pp. 1897–1882, 2001. DOI: 10.1016/s0031-3203(00)00150-3.

[43] A. L. Brun, A. S. Britto, L. S. Oliveira, F. Enembreck, and R. Sabourin, "Contribution of data complexity features on dynamic classifier selection," in *Proceedings of the International Joint Conference on Neural Networks*, Vancouver, Canada: IEEE, Oct. 2016, pp. 4396–4403, ISBN: 9781509006199. DOI: 10.1109/IJCNN.2016.7727774.

[44] M. Sabourin, A. Mitiche, D. Thomas, and G. Nagy, "Classifier combination for hand-printed digit recognition," in *Proceedings of the 2nd International Conference on Document Analysis and Recognition*, Tsukuba, Japan: IEEE, 1993, pp. 163–166. DOI: 10.1109/ICDAR.1993.395758.

[45] L. Bai, A. Velichko, and B. W. Drinkwater, "Characterization of defects using ultrasonic arrays: A dynamic classifier approach," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 62, no. 12, pp. 2146–2160, Dec. 2015, ISSN: 08853010. DOI: 10.1109/TUFFC.2015.007334.

[46] M. Goldstein and S. Uchida, "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data," *PLoS ONE*, vol. 11, no. 4, Apr. 2016. DOI: 10.1371/journal.pone.0152173.

[47] R. Gangireddy, "Related Problems," in *Knowing the Unknown: Open-World Recognition for Biodiversity Datasets (MSc thesis)*, Enschede, Apr. 2023.

[48] R. Chalapathy and S. Chawla, "Deep Learning for Anomaly Detection: A Survey," *Computing Research Repository*, vol. abs/1901.03407, 2019. DOI: 10.48550/arXiv.1901.03407. [Online]. Available: http://digital.library.wisc.edu/1793/60660.

[49] B. Settles, "Active Learning Literature Survey," University of Wisconsin, Wisconsin, Tech. Rep., Jan. 2009.

[50] B. Settles and M. Craven, "An Analysis of Active Learning Strategies for Sequence Labeling Tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ACL Press, 2008. DOI: 10.3115/1613715.1613855.

[51] E. B. Baum, "Neural Net Algorithms That Learn in Polynomial Time from Examples and Queries," *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 5–19, Jan. 1991, ISSN: 19410093. DOI: 10.1109/72.80287.

[52] X. Zhan, Q. Wang, K.-h. Huang, H. Xiong, D. Dou, and A. B. Chan, "A Comparative Survey of Deep Active Learning," Mar. 2022. DOI: 10.48550/arxiv.2203.13450.

[53] C. Shui, F. Zhou, C. Gagné, and B. Wang, "Deep Active Learning: Unified and Principled Method for Query and Training," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, Palermo, Italy: PMLR, Aug. 2020.

[54] S. Sinha, S. Ebrahimi, and T. Darrell, "Variational Adversarial Active Learning," in *International Conference on Computer Vision*, Seoul, South Korea: IEEE, Oct. 2019, pp. 5971–5980. DOI: 10.1109/ICCV.2019.00607.

[55] D. Gissin and S. Shalev-Shwartz, "Discriminative Active Learning," *Computing Research Repository*, vol. abs/1907.06347, 2019. DOI: 10.48550/arXiv.1907.06347.

[56] J. Azimi, A. Fern, X. Z. Fern, G. Borradaile, and B. Heeringa, "Batch Active Learning via Coordinated Matching," in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, United Kingdom, 2012. DOI: 10.48550/arXiv.1206.6458.

[57] K. Wei, R. Iyer, and J. Bilmes, "Submodularity in Data Subset Selection and Active Learning," in *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 2015.

[58] K. Woods, W. Philip Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997, ISSN: 01628828. DOI: 10.1109/34.588027.

[59] P. Du, J. Xia, W. Zhang, K. Tan, Y. Liu, and S. Liu, "Multiple Classifier System for Remote Sensing Image Classification: A Review," *Sensors*, vol. 12, no. 4, pp. 4764–4792, Apr. 2012. DOI: 10.3390/S120404764.

[60] P. K. Syriopoulos, S. B. Kotsiantis, and M. N. Vrahatis, "Survey on KNN Methods in Data Science," *Lecture Notes in Computer Science*, vol. 13621, Feb. 2022, ISSN: 16113349. DOI: 10.1007/978-3-031-24866-5_28.

[61] B. Li and L. Han, "Distance weighted cosine similarity measure for text classification," *Lecture Notes in Computer Science*, vol. 8206, 2013, ISSN: 03029743. DOI: 10.1007/978-3-642-41278-3_74.

[62] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining*, vol. 5, pp. 363–387, Oct. 2012, ISSN: 1932-1872. DOI: 10.1002/SAM.11161.

[63] A. Bendale and T. Boult, "Towards Open Set Deep Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1563–1572, Jun. 2016, ISSN: 10636919. DOI: 10.1109/CVPR.2016.173.

[64] S. Liang, Y. Li, and R. Srikant, "Enhancing the Reliability of Out-of-Distribution Image Detection in Neural Networks," in *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018. DOI: 10.48550/arXiv.1706.02690.

[65] I. J. Ndiour, N. A. Ahuja, and O. Tickoo, "Subspace Modeling for Fast Out-of-Distribution and Anomaly Detection," in *Proceedings of the International Conference on Image Processing*, Bordeaux, France: IEEE, Oct. 2022, pp. 3041–3045. DOI: 10.1109/icip46576.2022.9897694.

[66] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, Jan. 2016, ISSN: 1084-8045. DOI: 10.1016/J.JNCA.2015.11.016.

[67] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001, ISSN: 08997667. DOI: 10.1162/089976601750264965.

[68] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep Anomaly Detection with Outlier Exposure," in *Proceedings of the International Conference on Learning Representations*, New Orleans, USA, May 2019. DOI: 10.48550/arXiv.1812.04606.

[69] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi, "Generative OpenMax for Multi-Class Open Set Classification," *Computing Research Repository*, Jul. 2017. DOI: doi.org/10.5244/c.31.42.

[70] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection," in *Proceedings of the 3rd International Conference on Data Mining*, San Francisco, USA: SIAM, May 2003. DOI: 10.1137/1.9781611972733.3.

[71] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "An experimental evaluation of novelty detection methods," *Neurocomputing*, vol. 135, pp. 313–327, Jul. 2014, ISSN: 0925-2312. DOI: 10.1016/J.NEUCOM.2013.12.002.

[72] K. R. Shahapure and C. Nicholas, "Cluster quality analysis using silhouette score," in *Proceedings of the 7th International Conference on Data Science and Advanced Analytics*, Sydney: IEEE, 2020, pp. 747–748, ISBN: 9781728182063. DOI: 10.1109/DSAA49011.2020.00096.

[73] D. Hendrycks and K. Gimpel, "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks," in *Proceedings of the International Conference on Learning Representations*, Toulon, France, Apr. 2017. DOI: 10.1145/3503161.3548340.

[74] A. Nguyen, J. Yosinski, and J. Clune, "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2014, pp. 427–436. DOI: 10.1109/cvpr.2015.7298640.

[75] V. Škvára, T. Pevný, and V. Šmídl, "Are generative deep models for novelty detection truly better?" In *Workshop for the Special Interest Group on Knowledge Discovery and Data Mining*, London, Great Britain: ACM, Aug. 1932, ISBN: 9781450321389. DOI: 10.1145/1235.

[76] G. O. Campos, A. Zimek, J. Sander, *et al.*, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, pp. 891–927, Jul. 2016, ISSN: 1573756X. DOI: 10.1007/S10618-015-0444-8.

[77] A. Biswas and D. Parikh, "Simultaneous Active Learning of Classifiers & Attributes via Relative Feedback," in *Conference on Computer Vision and Pattern Recognition*, Portland, USA: IEEE, Jun. 2013, pp. 644–651. DOI: 10.1109/CVPR.2013.89.

[78] H. Ngo, M. Luciw, N. A. Vien, J. Nagi, A. Forster, and J. Schmidhuber, "Efficient Interactive Multiclass Learning from Binary Feedback," *Transactions on Interactive Intelligent Systems*, vol. 4, no. 3, Aug. 2014. DOI: 10.1145/2629631.

[79] E. Lucas, S. Whitaker, and T. C. Havens, "Online learning with binary feedback for multi-class problems," in *Symposium Series on Computational Intelligence*, Singapore: IEEE, Dec. 2022, pp. 374–380, ISBN: 978-1-6654-8768-9. DOI: 10.1109/SSCI51031.2022.10022153.

[80] W.-X. Bao, J.-Y. Hang, and M.-L. Zhang, "Partial Label Dimensionality Reduction via Confidence-Based Dependence Maximization," *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 46–54, Aug. 2021. DOI: 10.1145/3447548.3467313.

[81] S. He, L. Feng, F. Lv, W. Li, and G. Yang, "Partial-label Learning with Semantic Label Representations," in *Proceedings of the 28th International Conference on Knowledge Discovery and Data Mining*, vol. 22, Washington, USA: ACM, Aug. 2022, pp. 545–553, ISBN: 9781450393850. DOI: 10.1145/3534678.3539434.

[82] R. Bhattacharjee and N. Manwani, "Online Algorithms for Multiclass Classification Using Partial Labels," *Lecture Notes in Computer Science*, vol. 12084, pp. 249–260, May 2020, ISSN: 16113349. DOI: 10.1007/978-3-030-47426-3_20.

[83] H. Wang, M. Xia, Y. Li, *et al.*, "SoLar: Sinkhorn Label Refinery for Imbalanced Partial-Label Learning," in *Proceedings of the 36th Conference on Neural Information Processing Systems*, New Orleans, USA, 2022.

[84] C. Qiao, N. Xu, and X. Geng, "Decompositional Generation Process for Instance-Dependent Partial Label Learning," in *Proceedings of the International Conference for Learning Representations*, Kigali, Rwanda, May 2023. DOI: 10.48550/arXiv.2204.03845.

[85] L. Feng and B. An, "Partial Label Learning by Semantic Difference Maximization," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Macao, China, Aug. 2019, pp. 2294–2300. DOI: 10.24963/ijcai.2019/318.

[86] H. Wang, R. Xiao, Y. Li, *et al.*, "PiCO+: Contrastive Label Disambiguation for Robust Partial Label Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan. 2022. DOI: 10.1109/tpami.2023.3342650.

[87] M. Xu, Z. Lian, L. Feng, B. Liu, and J. Tao, "DALI: Dynamically Adjusted Label Importance for Noisy Partial Label Learning," Jan. 2023. DOI: 10.48550/arXiv.2301.12077.

[88] J. Fan, Y. Yu, and Z. Wang, "Partial Label Learning with competitive learning graph neural network," *Engineering Applications of Artificial Intelligence*, vol. 111, p. 104 779, Mar. 2022. DOI: 10.1016/J.ENGAPPAI.2022.104779.

[89] J. Fan, Y. Yu, Z. Wang, and J. Gu, "Partial Label Learning Based on Disambiguation Correction Net With Graph Representation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 8, pp. 4953–4967, Aug. 2022. DOI: 10.1109/tcsvt.2021.3139968.

[90] H. Wang, Y. Qiang, C. Chen, *et al.*, "Online Partial Label Learning," *Lecture Notes in Computer Science*, vol. 12458, pp. 455–470, Feb. 2021, ISSN: 16113349. DOI: 10.1007/978-3-030-67661-2_27.

[91] E. Hüllermeier and J. Beringer, "Learning from ambiguously labeled examples," *Intelligent Data Analysis*, vol. 10, no. 5, pp. 419–439, 2006, ISSN: 16113349. DOI: 10.1007/11552253_16.

[92] N. Nguyen and R. Caruana, "Classification with Partial Labels," in *Proceedings of the 14th International Conference on Knowledge Discovery & Data Mining*, Las Vegas, USA: ACM, Aug. 2008, pp. 551–559, ISBN: 9781605581934. DOI: 10.1145/1401890.1401958.

[93] M.-L. Zhang, F. Yu, and C.-Z. Tang, "Disambiguation-Free Partial Label Learning," *Transactions on Knowledge and Data Engineering*, vol. 29, no. 10, pp. 2155–2167, Oct. 2017. DOI: 10.1137/1.9781611973440.5.

[94] M.-L. Zhang, B.-B. Zhou, and X.-Y. Liu, "Partial Label Learning via Feature-Aware Disambiguation," in *Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining*, San Francisco, USA: ACM, Aug. 2016, ISBN: 9781450342322. DOI: 10.1145/2939672.2939788.

[95] L. Feng, J. Lv, B. Han, *et al.*, "Provably Consistent Partial-Label Learning," in *Proceedings of the 34th Conference on Neural Information Processing Systems*, Vancouver, Canada, Dec. 2020.

[96] J. Seo and J. S. Huh, "On the Power of Deep but Naive Partial Label Learning," in *International Conference on Acoustics, Speech and Signal Processing*, IEEE, Jun. 2021, pp. 3820–3824. DOI: 10.1109/icassp39728.2021.9414927.

[97] Y. Yan and Y. Guo, "Partial Label Learning with Batch Label Correction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 6575–6582, Apr. 2020, ISSN: 2374-3468. DOI: 10.1609/AAAI.V34I04.6132.

[98] N. Cebron, F. Richter, and R. Lienhart, ""I can tell you what it's not": active learning from counterexamples," *Progress in Artificial Intelligence*, vol. 1, pp. 291–301, Aug. 2012. DOI: 10.1007/s13748-012-0023-9.

[99] L. Feng, T. Kaneko, B. Han, G. Niu, B. An, and M. Sugiyama, "Learning with Multiple Complementary Labels," in *International Conference on Machine Learning*, Baltimore, USA, Jul. 2020.

[100] Y.-L. Chen, Y. F. Zheng, and Y. Liu, "Margin and Domain Integrated Classification for Images," *International Journal of Information Acquisition*, Apr. 2011. DOI: 10.1142/s0219878911002343.

[101] Y. Li, C. Liu, S. Zhao, and Q. Hua, "Active partial label learning based on adaptive sample selection," *International Journal of Machine Learning and Cybernetics*, vol. 13, pp. 1603–1617, Jun. 2022. DOI: 10.1007/S13042-021-01470-X.

[102] P. Hu, Z. C. Lipton, A. Anandkumar, and D. Ramanan, "Active Learning with Partial Feedback," in *Proceedings of the International Conference on Learning Representations*, New Orleans, United States, May 2019. DOI: 10.48550/arXiv.1802.07427.

[103] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. S. Yu, "Active Learning: A Survey," in *Data Classification: Algorithms and Applications*, CRC Press, 2014, ch. 22, ISBN: 9780429102639. DOI: 10.1201/b17320.

[104] P. Kumar and A. Gupta, "Active Learning Query Strategies for Classification, Regression, and Clustering: A Survey," *Journal of Computer Science and Technology*, vol. 35, no. 4, pp. 913–945, Jul. 2020. DOI: 10.1007/s11390-020-9487-4.

[105] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowledge and Information Systems*, vol. 35, no. 2, pp. 249–283, May 2013, ISSN: 02193116. DOI: 10.1007/S10115-012-0507-8.

[106] M. Wang and X.-S. Hua, "Active learning in multimedia annotation and retrieval," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 2, Feb. 2011, ISSN: 2157-6904. DOI: 10.1145/1899412.1899414.

[107] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Miami, USA: IEEE, Jun. 2009, pp. 2372–2379. DOI: 10.1109/CVPR.2009.5206627.

[108] F. Olsson, "A literature survey of active machine learning in the context of natural language processing," Swedish Intitute of Computer Science, Kista, Tech. Rep., Apr. 2009.

[109] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian Active Learning with Image Data," in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 2017.

[110] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by Committee," in *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, New York, United States: ACM, Jul. 1992, pp. 287–294. DOI: 10.1145/130385.130417.

[111] B. Settles, M. Craven, and S. Ray, "Multiple-Instance Active Learning," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, vol. 15, Vancouver, Canada: ACM, 1994, pp. 1289–1296.

[112] G. Qian, S. Sural, Y. Gu, and S. Pramanik, "Similarity between Euclidean and cosine angle distance for nearest neighbor queries," in *Proceedings of the Symposium on Applied Computing*, Nicosia, Cyprus: ACM, Mar. 2004, pp. 1232–1237. DOI: 10.1145/967900.968151.

[113] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, Jul. 2013, ISSN: 0022-1031. DOI: 10.1016/J.JESP.2013.03.013.

[114] J. Choi, I. Elezi, H. J. Lee, C. Farabet, and J. M. Alvarez, "Active Learning for Deep Object Detection via Probabilistic Modeling," in *Proceedings of the IEEE International Conference on Computer Vision*, IEEE, Oct. 2021, pp. 10 244–10 253, ISBN: 9781665428125. DOI: 10.1109/ICCV48922.2021.01010.

[115] T. H. Wu, Y. C. Liu, Y. K. Huang, *et al.*, "ReDAL: Region-based and Diversity-aware Active Learning for Point Cloud Semantic Segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, Montreal, Canada: IEEE, 2021, pp. 15 490–15 499, ISBN: 9781665428125. DOI: 10.1109/ICCV48922.2021.01522.

[116] O. D. Incel, M. Kose, and C. Ersoy, "A Review and Taxonomy of Activity Recognition on Mobile Phones," *BioNanoScience*, vol. 3, pp. 145–171, Jun. 2013, ISSN: 21911630. DOI: 10.1007/S12668-013-0088-3.

[117] M. V. Albert, S. Toledo, M. Shapiro, and K. Kording, "Using mobile phones for activity recognition in Parkinson's patients," *Frontiers in Neurology*, vol. 3, no. 158, Nov. 2012, ISSN: 16642295. DOI: 10.3389/FNEUR.2012.00158.

[118] *Keras: Deep Learning for humans*. [Online]. Available: https://keras.io/.

[119] L. Seidenari, V. Varano, S. Berretti, A. Del Bimbo, and P. Pala, "Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses," in *Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Portland, USA: IEEE, 2013, pp. 479–485, ISBN: 9780769549903. DOI: 10.1109/CVPRW.2013.77.

[120] V. Bloom, D. Makris, and V. Argyriou, "G3D: A gaming action dataset and real time action recognition evaluation framework," in *Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Providence, USA: IEEE, 2012, pp. 7–12, ISBN: 9781467316118. DOI: 10.1109/CVPRW.2012.6239175.

[121] M. Müller, A. Baak, and H.-P. Seidel, "Efficient and Robust Annotation of Motion Capture Data," in *Proceedings of the SIGGRAPH Symposium on Computer Animation*, E. Grinspun and J. Hodgins, Eds., ACM, 2009. DOI: 10.1145/1599470.1599473.

[122] S. Gaglio, G. L. Re, and M. Morana, "Human Activity Recognition Process Using 3-D Posture Data," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 5, pp. 586–597, Oct. 2015, ISSN: 21682291. DOI: 10.1109/THMS.2014.2377111.

[123] C. Chen, R. Jafari, and N. Kehtarnavaz, "UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor," in *Proceedings of the International Conference on Image Processing*, vol. 2015-December, Quebec, Canada: IEEE, 2015, pp. 168–172, ISBN: 9781479983391. DOI: 10.1109/ICIP.2015.7350781.

[124] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, Providence, USA: IEEE, 2012, pp. 1290–1297, ISBN: 9781467312264. DOI: 10.1109/CVPR.2012.6247813.

[125] V. T. Nguyen, T. N. Nguyen, T. L. Le, D. T. Pham, and H. Vu, "Adaptive most joint selection and covariance descriptions for a robust skeleton-based human action recognition," *Multimedia Tools and Applications*, vol. 80, pp. 27 757–27 783, Jul. 2021, ISSN: 15737721. DOI: 10.1007/S11042-021-10866-4.

[126] T. H. Tran, T. L. Le, D. T. Pham, *et al.*, "A multi-modal multi-view dataset for human fall analysis and preliminary investigation on modality," in *Proceedings of the International Conference on Pattern Recognition*, Beijing, China: IEEE, 2018, pp. 1947–1952, ISBN: 9781538637883. DOI: 10.1109/ICPR.2018.8546308.

[127] J. Wang, X. Nie, Y. Xia, Y. Wu, and S.-C. Zhu, "Cross-view Action Modeling, Learning and Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, USA: IEEE, Jun. 2014. DOI: 10.1109/cvpr.2014.339.

[128] S. Fothergill, H. M. Mentis, P. Kohli, and S. Nowozin, "Instructing people for training gestural interactive systems," in *Conference on Human Factors in Computing Systems*, Austin, USA: ACM, May 2012, pp. 1737–1746, ISBN: 9781450310154. DOI: 10.1145/2207676.2208303.

[129] A. d. S. Inacio, M. Gutoski, A. E. Lazzaretti, and H. S. Lopes, "OSVidCap: A Framework for the Simultaneous Recognition and Description of Concurrent Actions in Videos in an Open-Set Scenario," *IEEE Access*, vol. 9, pp. 137 029–137 041, Sep. 2021. DOI: 10.1109/access.2021.3116882.

[130] G. Garcia-Hernando, S. Yuan, S. Baek, and T. K. Kim, "First-Person Hand Action Benchmark with RGB-D Videos and 3D Hand Pose Annotations," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 409–419, Apr. 2017, ISSN: 10636919. DOI: 10.1109/CVPR.2018.00050.

[131] H. Duan, J. Wang, K. Chen, and D. Lin, "PYSKL: Towards Good Practices for Skeleton Action Recognition," in *Proceedings of the 30th International Conference on Multimedia*, ACM, Oct. 2022, pp. 7351–7354, ISBN: 9781450392037. DOI: 10.1145/3503161.3548546.

[132] MMAction2 Contributors, *OpenMMLab's Next Generation Video Understanding Toolbox and Benchmark*, Jul. 2020. [Online]. Available: https://github.com/open-mmlab/mmaction2.

[133] OpenMMLab, *Finetuning Models — MMAction2 1.1.0 documentation*, 2020. [Online]. Available: https://mmaction2.readthedocs.io/en/latest/user_guides/finetune.html.

[134] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 6, Jan. 2020, ISSN: 14712164. DOI: 10.1186/S12864-019-6413-7.

[135] D. Beckstein and K. Kaneda, "Manual Hyperparameter Tuning," in *Deep learning*, 7553, I. Goodfellow, Y. Bengio, and A. Courville, Eds., Cambridge: The MIT Press, 2016, ch. 11.4.1, pp. 423–426, ISBN: 9780262035613. [Online]. Available: http://www.deeplearningbook.org.

[136] M. Ranjit and G. Ganapathy, "A Study on the use of State-of-the-Art CNNs with Fine Tuning for Spatial Stream Generation for Activity Recognition," in *Proceedings of the 3rd International Conference on Electrical, Computer and Communication*

Technologies, Coimbatore, India: IEEE, Feb. 2019, ISBN: 9781538681572. DOI: 10.1109/ICECCT.2019.8869360.

[137] H. Kaur, H. S. Pannu, and A. K. Malhi, "A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions," *ACM Computing Surveys*, vol. 52, no. 4, Aug. 2019. DOI: 10.1145/3343440.

[138] O. Reyes, A. H. Altalhi, and S. Ventura, "Statistical comparisons of active learning strategies over multiple datasets," *Knowledge-Based Systems*, vol. 145, Jan. 2018, ISSN: 09507051. DOI: 10.1016/J.KNOSYS.2018.01.033.

[139] S. Liu, S. Xue, J. Wu, *et al.*, "Online Active Learning for Drifting Data Streams," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 186–200, Jan. 2023. DOI: 10.1109/TNNLS.2021.3091681.

[140] W. Xu, F. Zhao, and Z. Lu, "Active learning over evolving data streams using paired ensemble framework," in *Proceedings of the 8th International Conference on Advanced Computational Intelligence*, Chiang Mai, Thailand: IEEE, Feb. 2016, pp. 180–185, ISBN: 978-1-4673-7780-5. DOI: 10.1109/ICACI.2016.7449823.

[141] E. Garcia-Ceja and R. F. Brena, "Activity Recognition Using Community Data to Complement Small Amounts of Labeled Instances," *Sensors*, vol. 16, no. 6, p. 897, Jun. 2016. DOI: 10.3390/S16060877.

[142] M. Stikic, D. Larlus, and B. Schiele, "Multi-graph based semi-supervised learning for activity recognition," in *International Symposium on Wearable Computers*, Linz, Austria: IEEE, 2009, pp. 85–92. DOI: 10.1109/ISWC.2009.24.

[143] D. Mourtzis, E. Vlachou, V. Zogopoulos, *et al.*, "Customer feedback gathering and management tools for product-service system design," *Procedia CIRP*, vol. 67, pp. 577–582, 2018. DOI: 10.1016/J.PROCIR.2017.12.264.

Appendix A Toy Problem Variables

	Breakfast	Lunch	Dinner
Mean (temperature)	10	40	60
Variance (temperature)	10	10	10
Mean (Calories)	200	500	300
Variance (Calories)	100	50	50

Table 13: Variables used to generate the general dataset.

	Breakfast	Lunch	Dinner
Mean (temperature)	40	50	90
Variance (temperature)	10	10	10
Mean (Calories)	400	700	500
Variance (Calories)	100	50	50

Table 14: Variables used to generate the personal dataset with drift.

	Breakfast	Lunch	Dinner
Mean (temperature)	10	20	60
Variance (temperature)	10	10	10
Mean (Calories)	200	500	800
Variance (Calories)	100	50	50

Table 15: Variables used to generate the personal dataset with a class in an unexplored region.

	Breakfast	Lunch	Lunch
Mean (temperature)	10	40	60
Variance (temperature)	10	10	10
Mean (Calories)	200	500	300
Variance (Calories)	100	50	50

Table 16: Variables used to generate the personal dataset with combined classes.

Appendix B Labelling Task

B.1 Instructions

Both annotators received the same set of instructions as follows:

Please note:

- *You may assume the activity is listed. However, if you're sure that they are performing another activity, use the label "other" (e.g. "I'm sure that must be running instead of the activities listed"). If you simply have no idea what they are performing, use "I don't know".*
- *The activities are performed sitting on the ground, sitting on a chair and standing.*
- *Samples can be shaky or unclear.*
- *Restrictions:*
    1. *This person has lost their sense of smell and therefore has not sniffed/smelled during the data acquisition.*
    2. *This person has no stapler at home so cannot staple books.*

They could also ask questions about these instructions. Both asked what was meant with "samples can be shaky", so an elaboration was given that not all samples are comprehensible per se. If they encountered incomprehensible samples, they could just put "I don't know".

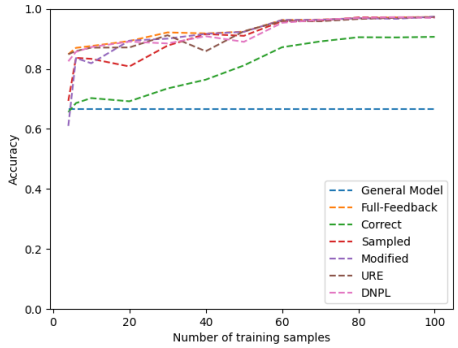
B.2 Observations

Both annotators found the labelling task relatively easy, while they both had no previous experience with annotating or working with skeleton data. This is an indication that activities could be recognized from skeleton data. However, one of the annotators also mentioned that they simply classified all activities with an arm moving upwards as *yawn* and all samples with arms being down as *cutting paper*. While this is a valid strategy for these classes, this would not be feasible if more classes were added. Additionally, one of the annotators also mentioned "I'm glad that sniff doesn't exist, because I wouldn't be able to distinguish them", where "them" refers to yawn and sniff. Therefore, while this task seems to be doable, it might be difficult to ask annotators to label a wide range of classes, where the movements are much more subtle.

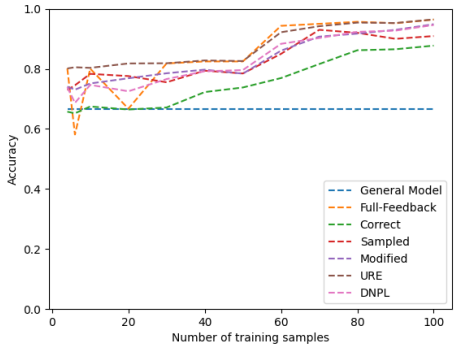
Appendix C Additional Results Toy Problem

C.1 Results without Framework

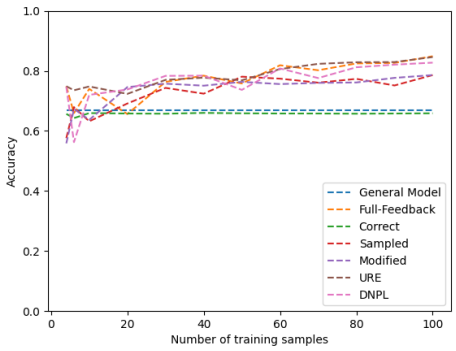
C.1.1 Limited Feedback



(a) Drift dataset



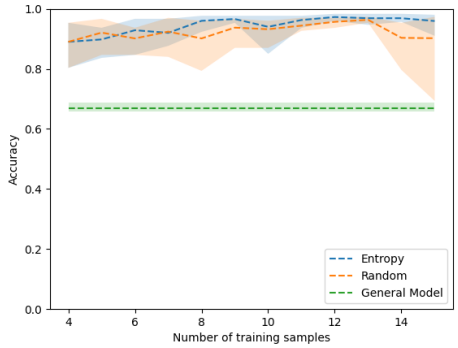
(b) Combined dataset



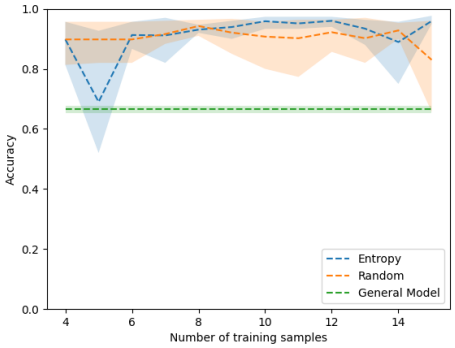
(c) Unexplored dataset

Figure 26: Mean accuracy with five-fold cross-validation of the LF techniques on the three toy problem datasets without framework, plotted per dataset.

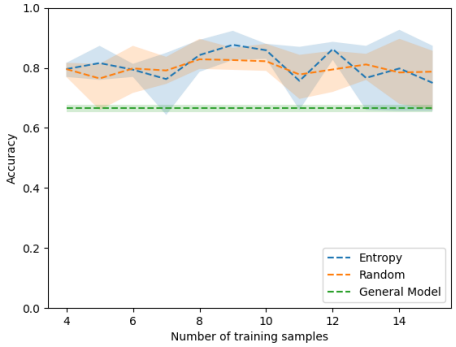
C.1.2 Active Learning



(a) Drift dataset



(b) Combined dataset



(c) Unexplored dataset

Figure 27: Performances of the *URE* with entropy-based and random sampling on the three toy problem datasets without the framework.

C.2 Framework Comparison Personal Model

		Personal Model					
		Breakfast		Lunch		Dinner	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Personal model with framework	Correct	515	0	465	0	475	0
	Incorrect	10	25	0	10	0	0

(a) Drift

		Personal Model			
		Breakfast		Lunch	
		Correct	Incorrect	Correct	Incorrect
Personal model with framework	Correct	520	0	910	0
	Incorrect	0	30	0	40

(b) Combined[56]

		Personal Model					
		Breakfast		Lunch		Dinner	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Personal model with framework	Correct	535	0	330	105	435	0
	Incorrect	5	10	0	40	0	40

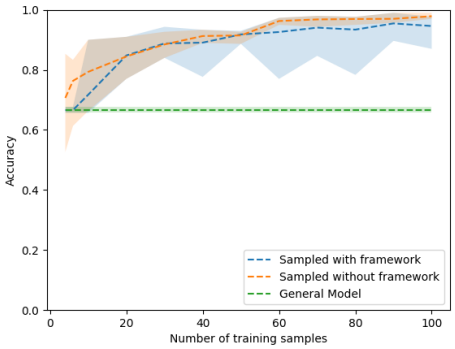
(c) Unexplored

Table 17: Comparison between the number of correctly and incorrectly predicted samples from the personal model and personal model with the framework (PM+F), for the (a) drift, (b) combined and (c) unexplored datasets.

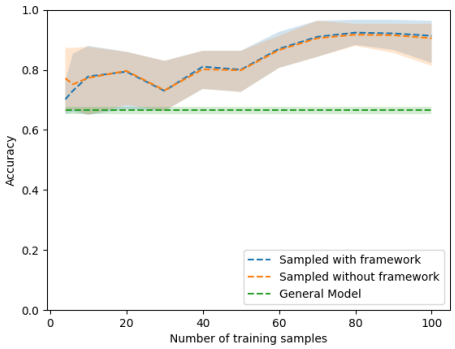
[56] Please note that the test set of the combined dataset does not contain any samples from the *dinner* class, which is therefore not included in this table.

C.3 Limited Feedback Variation

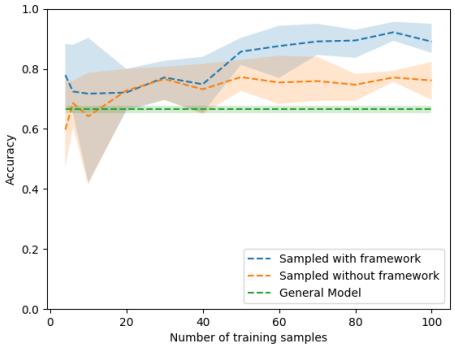
This subsection contains the variation of the limited feedback techniques *Sampled*, *Modified*, *DNPL* and *URE*. The variation of *Correct* can be found in Figure 17 and the variation of the *Full-Feedback* model can be found in Figure 15[57].



(a) Drift dataset



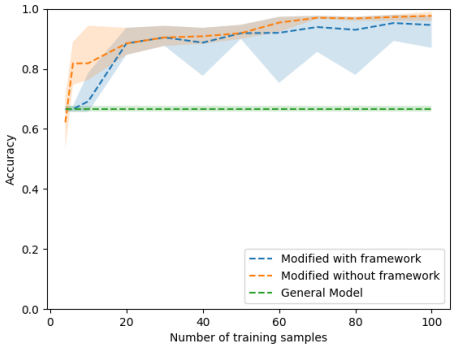
(b) Combined dataset



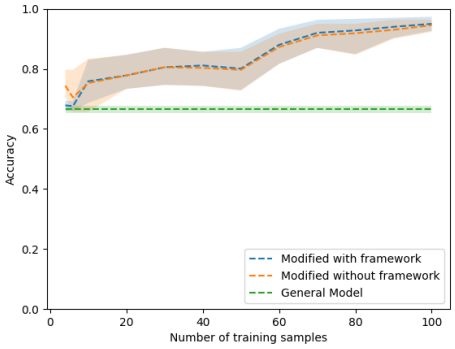
(c) Unexplored dataset

Figure 28: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) for the drift, unexplored and combined datasets for the *Sampled* method with five-fold cross-validation.

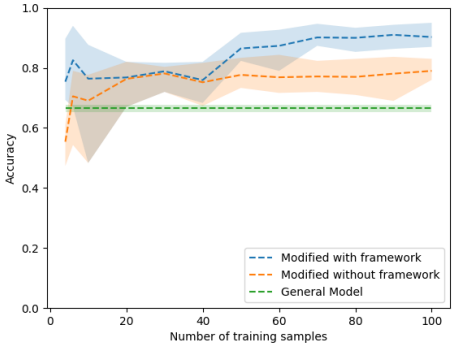
[57]The personal model in the figure uses full-feedback, and is, therefore, the same as the *Full-Feedback model* used as a reference to the LF techniques.



(a) Drift dataset

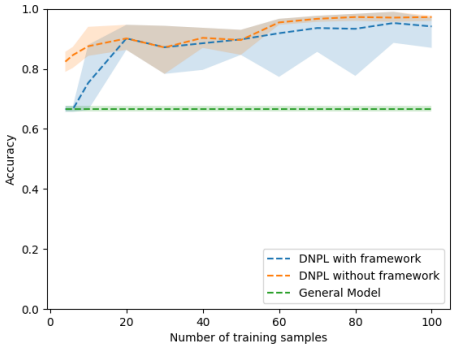


(b) Combined dataset

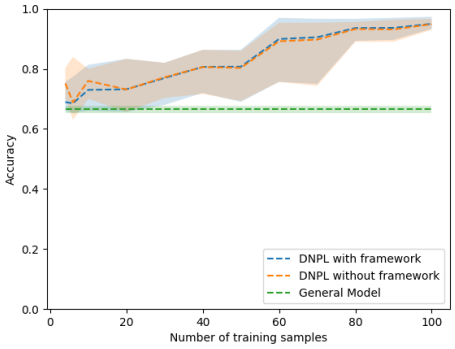


(c) Unexplored dataset

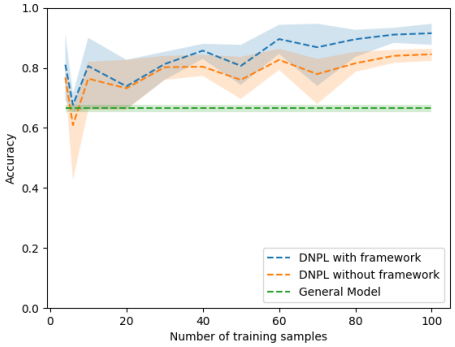
Figure 29: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) for the drift, unexplored and combined datasets for the *Modified* method with five-fold cross-validation.



(a) Drift dataset

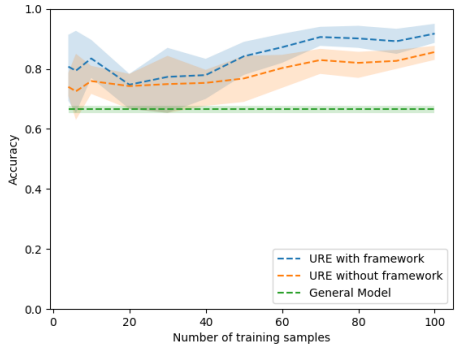


(b) Combined dataset

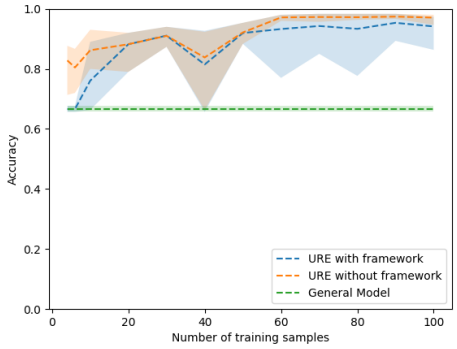


(c) Unexplored dataset

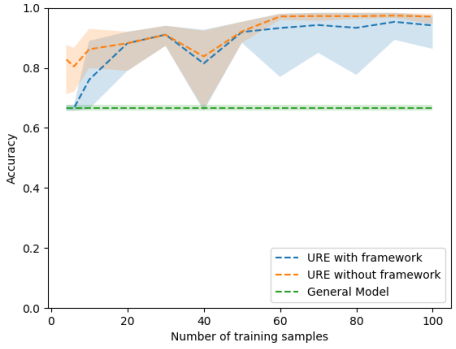
Figure 30: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) for the drift, unexplored and combined datasets for the *DNPL* method with five-fold cross-validation.



(a) Drift dataset



(b) Combined dataset



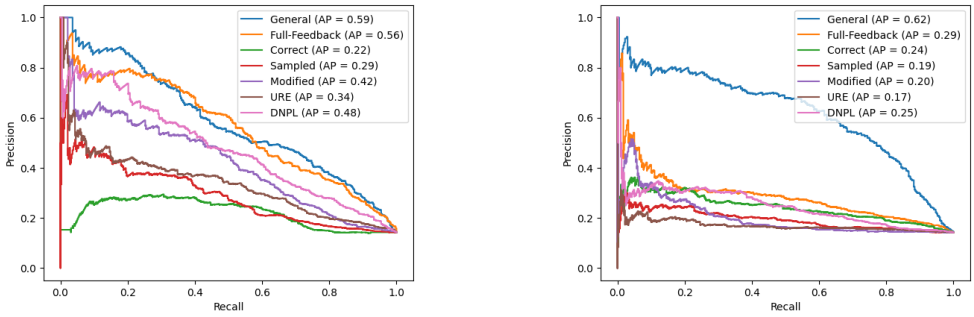
(c) Unexplored dataset

Figure 31: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) for the drift, unexplored and combined datasets for the *URE* method with five-fold cross-validation.

Appendix D Additional Results HAR

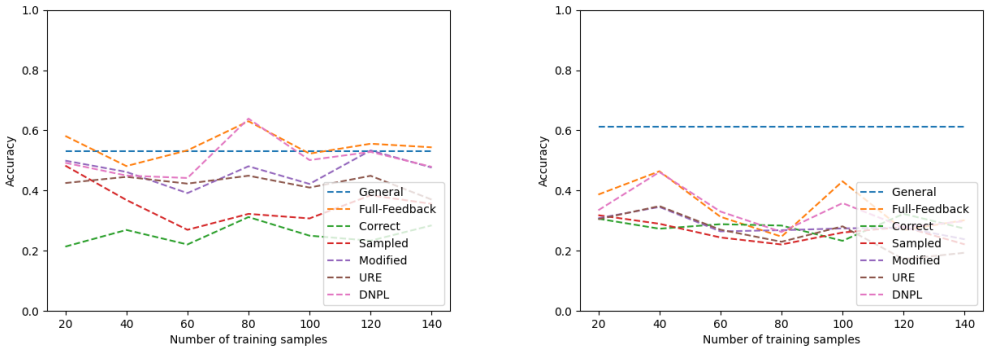
D.1 Results without Framework

D.1.1 Limited Feedback



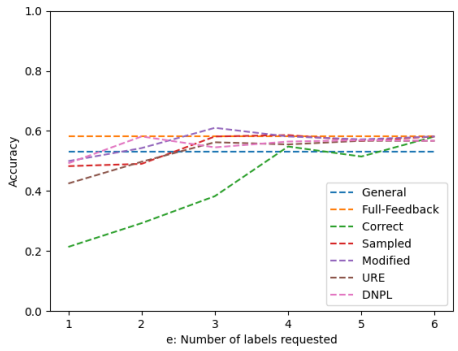
(a) P008 (b) P041

Figure 32: Precision-recall curves of the limited feedback methods *Correct*, *Sampled*, *Modified*, *URE* and *DNPL* in comparison to the *Full-Feedback* model and general model with $e = 1$ for (a) P008 and (b) P041.

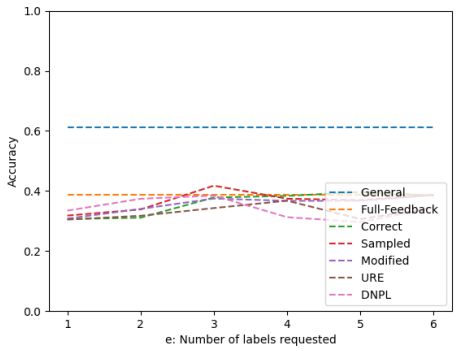


(a) P008 (b) P041

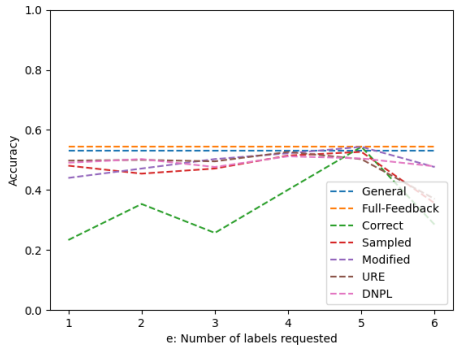
Figure 33: Performance of the limited feedback methods *Correct*, *Sampled*, *Modified*, *URE* and *DNPL* in comparison to the *Full-Feedback* model, without framework on the subjects (a) P008 and (b) P041 from the HAR dataset.



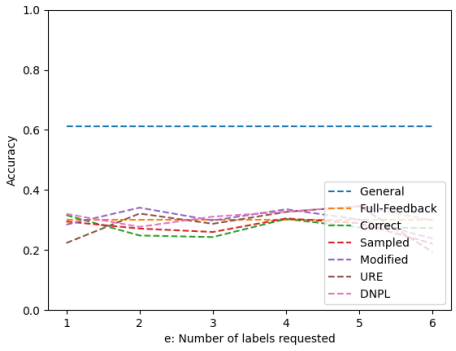
(a) $|T| = 20$, P008



(b) $|T| = 20$, P041



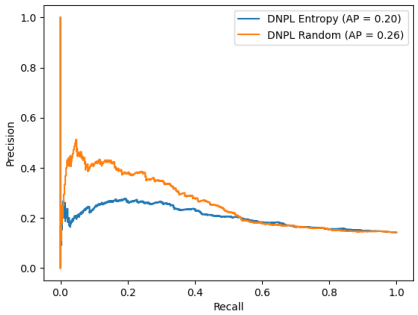
(c) $|T| = 140$, P008



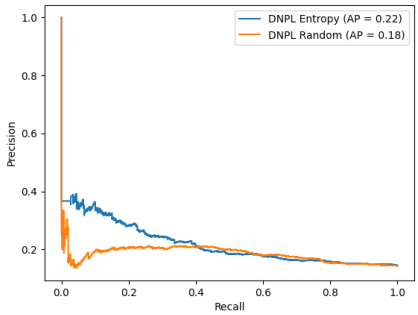
(d) $|T| = 140$, P041

Figure 34: Mean accuracy of the limited feedback methods *Correct*, *Sampled*, *Modified*, *URE* and *DNPL* in comparison to the *Full-Feedback* model and general model with e varied between 1 and 6 with five-fold cross-validation, with $T = 20$ and $T = 140$ for P008 and P041, without framework.

D.1.2 Active Learning

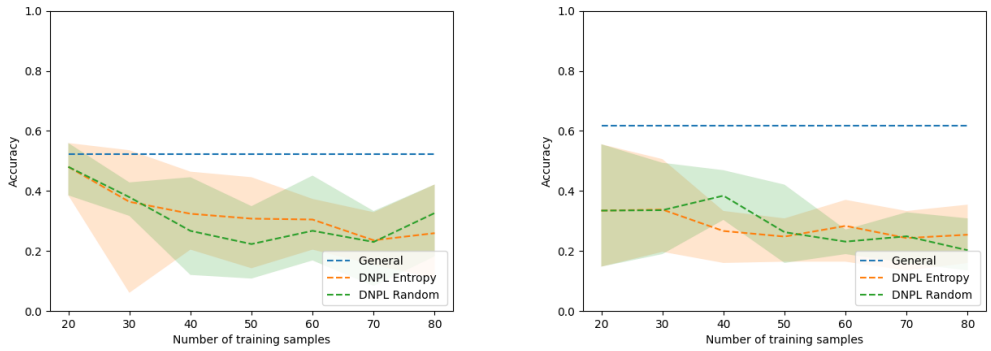


(a) P008



(b) P041

Figure 35: Precision-recall curves of random sampling and entropy-based sampling with *DNPL* with $e = 1$ for (a) P008 and (b) P041.



(a) P008 (b) P041

Figure 36: Mean accuracy of entropy-based and random sampling with *DNPL* with $e = 1$ for (a) P008 and (b) P041 without framework.

D.2 Framework Comparison Personal Model

		Personal model									
		Yawn		Cutting paper		Cutting nails		Put on headphone		Other	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
PM+F	Correct	58	9	91	2	18	7	34	17	17	7
	Incorrect	4	31	2	39	0	35	0	9	2	34

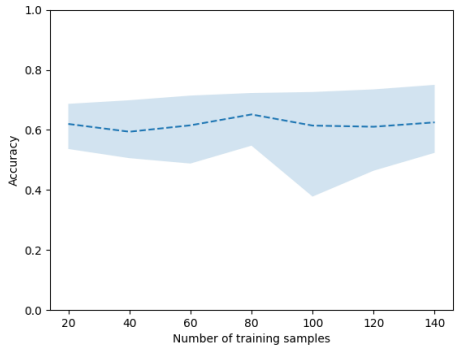
(a) P008

		Personal model									
		Yawn		Cutting paper		Cutting nails		Put on headphone		Other	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
PM+F	Correct	22	21	48	16	12	24	20	23	12	16
	Incorrect	0	58	5	47	1	23	0	17	0	33

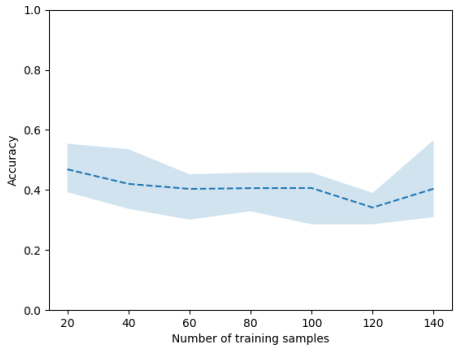
(b) P041

Table 18: Comparison between the number of correctly and incorrectly predicted samples from the personal model and personal model with the framework (PM+F) for (a) P008 and (b) P041.

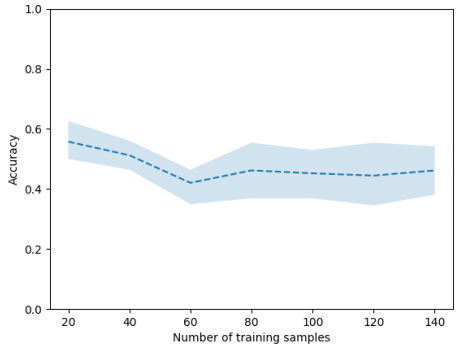
D.3 Limited Feedback Variation



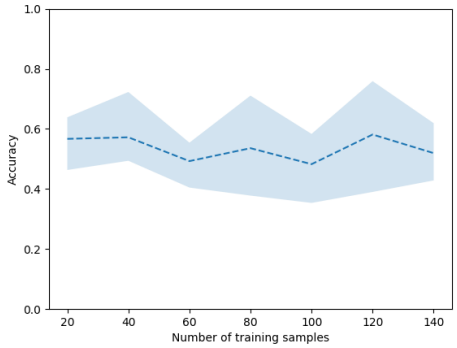
(a) *Full-Feedback*



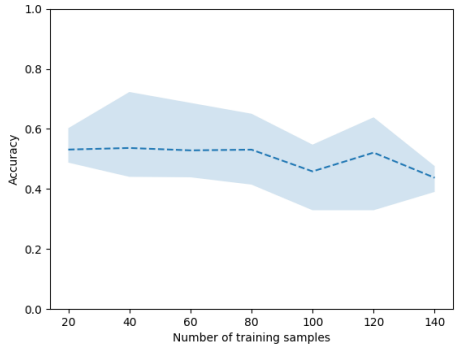
(b) *Correct*



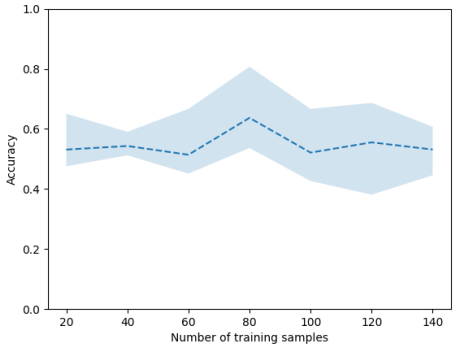
(c) *Sampled*



(d) *Modified*

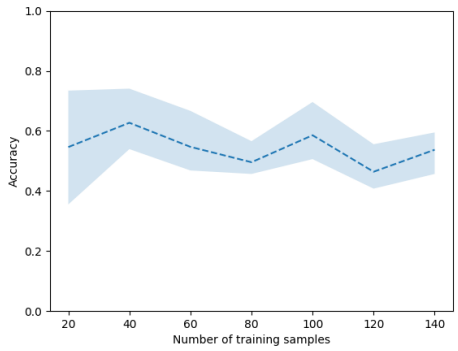


(e) *URE*

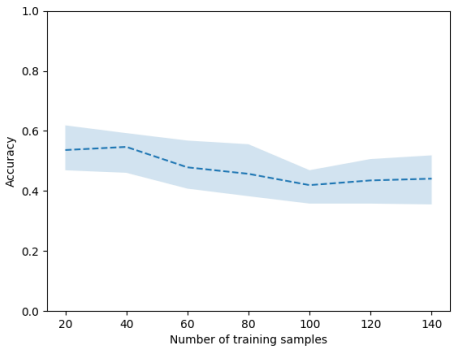


(f) *DNPL*

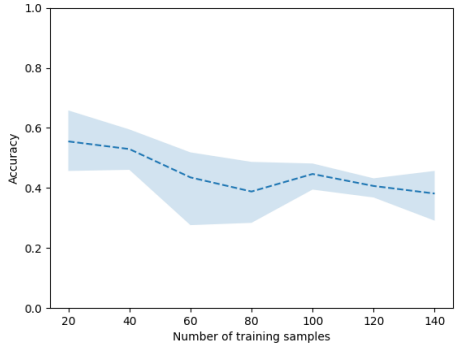
Figure 37: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) of (a) *Full-Feedback*, (b) *Correct*, (c) *Sampled*, (d) *Modified*, (e) *URE* and (f) *DNPL* with the framework for P008.



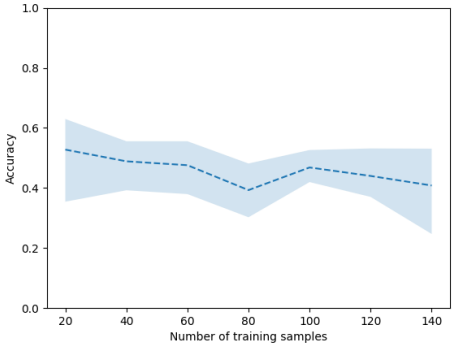
(a) *Full-Feedback*



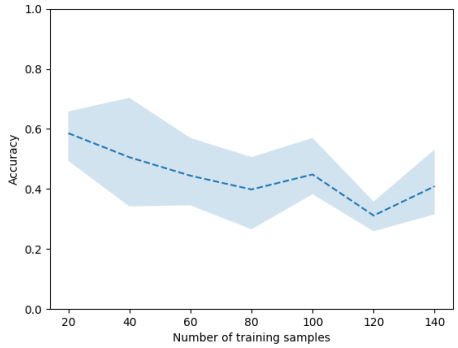
(b) *Correct*



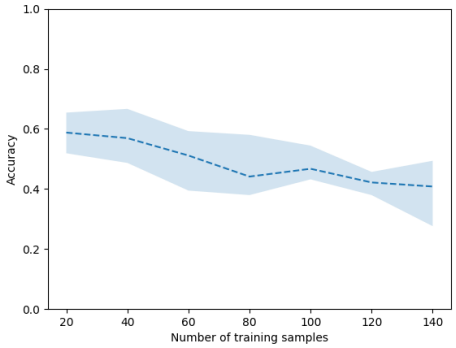
(c) *Sampled*



(d) *Modified*



(e) *URE*



(f) *DNPL*

Figure 38: Mean accuracy (dotted line) and minimum and maximum accuracy (filled area) of (a) *Full-Feedback*, (b) *Correct*, (c) *Sampled*, (d) *Modified*, (e) *URE* and (f) *DNPL* with the framework for P041.