SALTS: Streamlined Adaptive Learning for Sensors Time Series

Sotirios Vavaroutas[1], Georgios Rizos[1], Cecilia Mascolo[1]

***Abstract*— Sensor-generated time series hold immense potential across the healthcare domain, yet present challenges in labelling due to their sequential nature, which requires consideration of context and temporal dependencies. Recognising the costly nature of data labelling and that domain experts may have limited technical expertise in model optimisation, we introduce an approach to automate machine learning model training for medical time series, enhancing analysis efficiency.**

Our proposal first operates at the data input level via adaptive data acquisition, facilitating the selection of highly-informative samples for labelling. Further, it works at the model level, through dynamic model refinement to optimise the model on-the-fly by progressively exploring the possible hyperparameter options and choosing the best combination at each acquisition step, and through an automatic learning phase to maximise the usage of any unlabelled samples. This results in a robust learning strategy that continuously refines the model with expanding data and human expertise. Demonstrated on EEG, ECG, and IMU health signal classification, our method outperforms baselines and the current state-of-the-art, while reducing reliance on human input for model tuning. SALTS enhances the applicability of machine learning to healthcare time series, maximising the information gained through each human annotation step in an automated way.

***Keywords*— Semi-Supervised Learning, Active Learning, Dynamic Data Acquisition, Hyperparameter Optimisation**

I. Introduction

ML models are increasingly popular in healthcare, where they support personalised monitoring, diagnostics, and performance tracking. However, medical experts face significant challenges due to the need for large amounts of labelled data. Sensors data are frequently unlabelled [1] and thus require effort to prepare. This often acts as a bottleneck to the adoption of ML in biosignal applications, as the ML workflow from data annotation to model design and training is not sufficiently abstracted and automated. Labelling raw medical data and integrating them into the ML pipeline is a laborious process that needs close supervision by experts [2].

Prior work has explored human-in-the-loop strategies for making the most of a human annotation budget through Active Learning (AL) [1], [3]. However, such approaches often assume a fixed hyperparameter (HP) selection occurring before the data labelling, or a tuning step that remains static throughout the acquisition process: both result in suboptimal performance due to their inability to adapt hyperparameters during the acquisition. Additionally, an approach that aims to utilise information from unlabelled data, semi-supervised learning, often does not fully leverage domain expertise, overlooking the knowledge that medical experts can introduce [4]. Active semi-supervised learning [5] offers a step towards addressing this, though the need for manually tuning model hyperparameters at each data acquisition introduces complexity in addition to the laborious data labelling [2].

Healthcare time series, unlike other data modalities where labelling can be relatively straightforward, often pose a challenge in annotation, as this requires considerable medical expertise from labellers [6]. Their sequential nature also complicates the labelling [7], leading to bottlenecks in model development. This makes it essential to automate the ML pipeline, focusing on efficiently labelling high-value data while leveraging the inherent structure of unlabelled data.

We introduce SALTS, an end-to-end training solution for healthcare time series that makes the most of a user's annotation budget. We first adopt an adaptable data acquisition strategy to interactively prompt users to selectively label new data [6]. At the same time, we delegate hyperparameter tuning [8] to a dedicated refinement step, to automate the key task of choosing the most informative hyperparameters of the learning process. Subsequently, we use an automated training phase inspired by semi-supervised learning [4] to enhance predictions by taking into account the latent manifold on which the data lie through generating pseudo-labels for unlabelled samples, allowing the model to self-finalise. SALTS offers greater automation compared to baselines, and only requires initial dataset labelling from users.

Our main contributions are summarised as follows:

- We bring hyperparameter refinement concepts to an AL-inspired adaptive data acquisition process, by automatically tuning model hyperparameters iteratively with the data labelling. This requires much less human input pertaining to the model design in an AL setting.
- Our technique makes the most of a pre-set annotation budget, by having progressively more highly-tuned versions of the model being queried for informative data. This reduces the burden on data labellers, while maximising the value extracted from each labelled instance.
- We use unlabelled data through semi-supervised learning after training on as many labelled data as possible, enhancing performance with no additional labelling.
- We demonstrate SALTS in both binary and multi-class time series classification, using electroencephalography (EEG), electrocardiogram (ECG), and inertial measurement unit (IMU) signals. Compared to alternatives, it brings improvements up to 9% in accuracy in EEG data, up to 6% in ECG data, and up to 22% in IMU data.

II. Related Work

Active Learning. Advances in AL literature enable algorithms to achieve higher levels of accuracy given a constrained annotation budget, by iteratively selecting the data

[1]University of Cambridge

points that need to be labelled [6]. Prior work has examined human-in-the-loop strategies in medical signal classification for ECG and EEG recordings [9] and human activity recognition (HAR) [3], showcasing its efficacy in reducing annotation costs. However, the state-of-the-art cannot integrate human input in the labelling without pre-configured model hyperparameters, nor can it leverage unlabelled data.

Semi-Supervised Learning. Towards further incorporating information from unlabelled time series, the area of semi-supervised learning [4] is of interest. This can leverage the underlying manifold of unlabelled samples to improve model performance and generalisation. It, thus, makes the most of manually-labelled data, rendering it a cost-effective and scalable approach for model training on unlabelled data. Prior work has highlighted the pivotal role of semi-supervised learning in human activity recognition (HAR) tasks [10], but while this effectively articulates the concept, it may not capture the expertise of medical professionals. This is something that only active semi-supervised learning has achieved to date [11], which combines the power of AL and semi-supervised learning for better results. Both concepts are designed to work in the presence of unlabelled data [12], yet overlook the challenge of hyperparameter tuning. Consequently, users face the cumbersome task of selecting the best model hyperparameters by just picking a set a-priori, which adds complexity to the already-intricate process of data labelling. This underscores the necessity of not only exploring the hyperparameter space systematically, but also incorporating both human and model insights.

Hyperparameter Optimisation. This is a key aspect of AutoML [13] and can automatically select the best parameters for a given model in a principled manner, tailoring them to varying tasks [8]. The use of AutoML has been explored for time series [14] and for healthcare use cases [15], yet the hyperparameter tuning of the AL process remains under-explored, especially as AL introduces further complexity to a domain expert like a practitioner that may have little ML knowledge. Automating ML stages otherwise needing human intervention, like hyperparameter tuning, is imperative.

AutoDAL. Prior literature looking at boosting AL with hyperparameter tuning is similarly limited. The most relevant study is AutoDAL [16], which is targeted at a distributed computing setup, and involves partitioning data across machines, building independent models, and independently querying for labels based on partial views of the data, thus departing from the generality of the AL task. It also uses a costly regularisation that is based on a radial basis function kernel, whose hyperparameters are iteratively tuned using a combination of a genetic algorithm [17] and a generating set search [18]. Instead, our approach is based on Bayesian optimisation [19] and is able to tune a wider range of hyperparameters, including the discrete sets of activation functions and optimisers [20]. Moreover, calculating the kernel similarity matrix in AutoDAL is of $O(n^2)$ space and time complexity, where n is the dataset size. This is not scalable, and means that for the ECG dataset (see Section IV), AutoDAL would need over 70GB of GPU memory to run on a single worker machine. This shows the degree to which AutoDAL is tied to a multi-worker approach, which limits its generality. Its authors do not offer an implementation of AutoDAL, but we use the largest dataset that they use to perform a direct comparison on ECG sensor signal classification. In this comparison, our method outperforms AutoDAL, achieving a notable improvement in accuracy with the same amount of labelled samples.

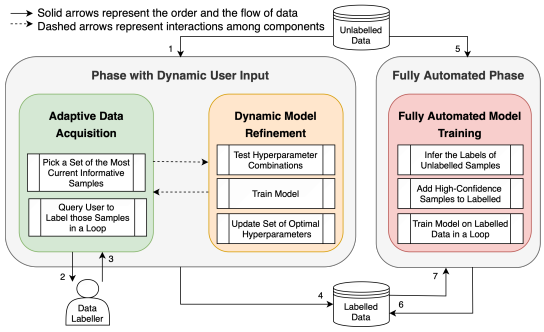

Fig. 1. Overview of SALTS. A portion of the most uncertain data is labelled at the adaptive data acquisition phase, while the model's hyperparameters are continuously refined. Then, a fully automated phase takes over, where the model pseudo-labels remaining samples, and incorporates them for training.

Weak Supervision. Other work using unlabelled data and human annotations, has looked at weak supervision to automatically generate training samples from medical data [21]. Work leveraging noisy heuristics and distant supervision [22] has also combined labelling functions and probabilistic label aggregation, which iteratively refines the labelling functions. However, designing effective labelling functions and handling the probabilistic nature of the labels can be challenging [23]. Additionally, the reliance on weakly supervised labels may introduce substantial noise [23], which can be detrimental in health data. The effort to develop labelling functions for biosignals often outweighs the savings in manual labelling, making the approach less effective.

III. The SALTS Framework

In this section we discuss our proposed methodology. This features an adaptive training phase operating on an initial small part of the available data by requesting the relevant labels by the user, and an automated part that operates subsequently on unlabelled data with no further user input.

A. System Overview

Dynamic User Input Phase. SALTS begins with an adaptive data acquisition, based on the pervasive assumption in AL literature that there is an initial core set that is already labelled [24], [6]. Once a limited number of samples has been labelled by the user, a preliminary HP selection phase refines the model configuration. The AL loop then smartly selects samples for labelling through uncertainty sampling, employing hyperparameters from the last optimisation pass. This aims to reduce the required human effort by delegating parts of the process to the automated algorithm. This process continues until the maximum annotation budget is expended, while also running a set of search space exploration trials in between the label acquisition iterations, ensuring that the hyperparameters of the model are continuously tuned.

Data: Unlabelled data D_U, initial hyperparameters H
Output: Labelled data D_L, trained model M_L
$D_{temp} \leftarrow$ small initial part of D_U
$D_{temp} \leftarrow$ request user to label D_{temp}
$D_L \leftarrow D_L \cup D_{temp}$
$H \leftarrow$ average values in range of each hyperparameter
$M_L \leftarrow$ model.train(D_L)
while *hyperparameter trials* $<$ *initial trials limit* **do**
  $H \leftarrow$ use M_L to explore search space
end
while *active learning iteration* $\leq$ *labelling threshold* **do**
  $D_{temp} \leftarrow$ most informative small part of D_U
  $D_{temp} \leftarrow$ request user to label D_{temp}
  $D_L \leftarrow D_L \cup D_{temp}$
  $M_L \leftarrow$ model.train(D_L)
  while *hyperparameter trials* $<$ *per-loop limit* **do**
    $H \leftarrow$ use M_L to explore search space
  end
end
while *semi-supervised iteration* $\leq$ *threshold* **do**
  $D_{temp} \leftarrow$ use M_L to infer labels for samples in D_U and pick those with a high confidence
  $D_{temp} \leftarrow$ automatically pseudo-label D_{temp}
  $D_L \leftarrow D_L \cup D_{temp}$
  $M_L \leftarrow$ model.train(D_L)
end

Algorithm 1: Proposed SALTS Approach.

Automated Training Phase. This commences without an explicit initialisation, utilising the data samples that have previously been labelled in the dynamic input phase. Pseudo-labelling is introduced to perform an inference pass on the unlabelled samples and to automatically assign labels to high-confidence predictions on those unlabelled samples [4]. The model is then trained on both the pseudo-labelled and the previously-labelled data, facilitating the integration of valuable information from unlabelled samples that an AL solution alone would not be able to capture. This strikes a balance between user involvement and automated processes.

B. Adaptive Data Labelling and Hyperparameter Tuning

To start the training, our system begins by selecting a small portion of the training data at random and queries the oracle for providing the labels for those samples, inspired by the common step that applies when using AL [24], [6]. The oracle refers to the human-in-the-loop doing the labelling [6], which in our case is performed by an automatic system for testing purposes as we have access to both the labelled and the unlabelled versions of the datasets for our case studies.

Data Labelling. Following the initial labelling, the learner in each iteration uses uncertainty sampling [25], enabling progressively more informed decisions on which samples to query the user for labelling next. The uncertainty of each classification using this method is defined by $1 - P(\hat{x} \mid x)$, where x is the instance to be predicted and $\hat{x}$ is the most likely prediction. While we employ uncertainty sampling in this study, the AL acquisition method that can be used by SALTS is flexible and could be adapted to other selection strategies. Similarly, any suitable model can be integrated within our SALTS framework. Once the user annotates a batch of samples, this batch is added to the labelled data pool, and the process repeats in subsequent iterations. A generic AL approach would continue this cycle [6]; however, our framework augments the process by tuning the model hyperparameters, allowing the model to be fine-tuned.

Hyperparameter Space Exploration. To tune the hyperparameters, we employ Bayesian optimisation [19], which systematically explores the hyperparameter space by leveraging information from prior trials to guide subsequent ones [20]. Unlike alternatives like random search that focus on multiple local optima, Bayesian optimisation aims to identify optimal hyperparameters more reliably by focusing on promising regions of the search space, often resulting in improved model performance consistency [19]. This provides a structured alternative to less guided search strategies.

The hyperparameters we tune in our solution are strategically diverse. Specifically, the tuner automatically picks the best Conv1D activation function amongst options including eLU [26], ReLU [27], Leaky-ReLU [28] (which facilitates a small gradient for more advanced learning), SeLU (Scaled eLU) [29], and GeLU (Gaussian eLU) [30]. The tuning then extends to the pool size of the MaxPool1D layer, tailoring the network's ability to extract meaningful features. It also tunes the activation function of the dense layer amongst ReLU [27], Sigmoid [31], and TanH, offering flexibility in shaping the network's response to different data characteristics. The optimisation process also selects the most suitable optimiser from a spectrum including Adam [32], AdaDelta [33], AdaGrad [34], AdaMax [32], and RmsProp.

C. Automated Training on Unlabelled Data

Following the adaptive training phase, our solution stops querying for more labels. At this point, it has learned enough information such that it can confidently classify unlabelled samples. Thus, to train on as many remaining samples as possible, our system proceeds with self-training, a widely-used type of semi-supervised learning [4]. This step has no initialisation phase, since a key part of the dataset has already been annotated. Using pseudo-labelling, SALTS picks high-confidence predictions and auto-assigns to them the relevant classification labels. For this, we follow the common case of letting the pseudo-labelling run for up to 20 iterations for convergence [35], [36], or until no more high-confidence predictions are available (whichever comes first). This lets the algorithm to train on the pseudo-labelled samples and then utilise its knowledge to pseudo-classify further samples [4].

D. Focal Loss for Addressing Data Imbalance

We use a focal loss function [37] to address class imbalance in the data, as it is designed to emphasise the harder, misclassified samples at each epoch, potentially also mitigating bias by skewed class distributions [38]. We implement focal loss using the relevant PyPI TensorFlow package [39] to better use the unlabelled data. Focal loss operates by adding a modulating factor $(1 - p_t)^\gamma$ to a cross-entropy loss, in order to dynamically scale it and focus on hard samples [38]. When it uses $\gamma = 0$, then it behaves in the same way as a cross-entropy loss. To automate the process of selecting γ, in SALTS we set the HP tuner to automatically optimise this value and ensure that the scaling factor can rapidly let the model focus on informative samples.

IV. Case Studies

In this section, we present the datasets used in our evaluation. Each dataset is diverse within the health domain, and classification is performed on a sample-by-sample basis, treating each sample independently. The raw time series signals were utilised as provided, thus incorporating preprocessing steps applied by the original dataset creators, but without any further denoising or feature engineering steps.

EEG Signal Classification. For our first application, we test our proposed method on electroencephalography (EEG) health data. We use the Epileptic Seizure Recognition (ESR) dataset of the UCI ML Repository [40], which is a dataset of EEG recordings modelled as time series. It features 11 500 labelled samples, with each one of them containing 178 attributes. The labels attached to each sample identify if the subject suffered from epileptic seizure or not, further classifying the cases in which the subjects did not suffer in four classes depending on whether they had their eyes open or closed during the data collection process for instance. As the boundaries among the four non-epileptic classes are insignificant, this dataset is used for binary classification, classifying the epileptic seizure class against the rest [40].

ECG Signal Classification. For our second application, we use an electrocardiogram (ECG) heartbeat categorisation dataset [41], [42] from the PhysioNet MIT-BIH Arrhythmia database [43]. This contains 109 446 samples and is characterised by a sampling frequency of 125Hz, with each signal segmented into 188 points. Its biosignals encompass diverse cardiac events, classified into the classes of normal, supraventricular, ventricular, fusion, and unknown beats.

IMU Signal Classification. For our third application, we evaluate on Inertial Measurement Unit (IMU) health data for human activity recognition. Specifically, we use the accelerometer time series of the Motion Sense dataset [44], which were captured using the accelerometer sensors of a smartphone. This dataset consists of recordings collected from 24 participants who represented diverse demographics, and who were asked to perform 6 activities across 15 trials. While the measurements are associated with specific classes based on the activities performed, it is important to note that they do not involve traditional human annotation. Instead, each participant is assigned a class label based on their activity during the data collection. Despite this distinction, we include this dataset to simulate the AL process on a further scenario. The samples have a sampling rate of 50Hz, and we use them for multi-class classification to distinguish activities between six classes: walking, walking downstairs, walking upstairs, jogging, sitting, and standing.

V. Experimental Setup

We now discuss our experimental setup, baselines, and the results in each case. For our evaluation, we run three repeated trials for each experiment and report their average values.

A. Baselines

To test the efficacy of SALTS we establish some critical baselines: an active learning baseline, a semi-supervised learning baseline, an AutoML baseline, and a hyperparameter tuning baseline. AL strategically selects informative instances for user annotation, expanding the labelled set in an intelligent manner, while semi-supervised learning leverages both user-labelled and unlabelled data to enhance model performance. On the other hand, both AutoML and hyperparameter tuning ensure that the model's configuration is tuned, but they require labelled data to operate. Across all baselines and our system, we run two sets of experiments: one where we label 10% of the samples, and one where we label 20%. These values have been chosen to showcase the alternatives' behaviour at different labelled data percentage allowances, while relevant work like AutoDAL [16] uses the same maximum labelling percentage of 20%. By employing these baselines and providing them all with the same percentage of user-labelled samples, we aim to assess our method's contribution with the same consistent input.

1) Active Learning Baseline: This baseline uses a standard AL paradigm, and is built using the modAL framework [45]. For its query strategy, we have chosen an uncertainty-based sampling method [25] that is the same as the one used in SALTS, towards a fair comparison. For the AL implementation, we have set the learner to initially ask the user to annotate an arbitrarily-chosen 1% of the samples in each dataset (seed 111), and then to query them for more labels in a loop until it is fitted to a maximum of 10% or 20% of the dataset (in different experiments). At each iteration, the learner chooses 1% of the samples, corresponding to the most uncertain each time, and the maximum percentage is set so that user does not annotate more samples with AL than with the other baselines. The shortcomings of this baseline are that the user needs to manually set the model hyperparameters to fine-tune them for each task, and that the model is only fitted to as many samples as the user chooses to label (i.e., 10% or 20% in our case). For these AL experiments, we use the same model hyperparameters as in the semi-supervised learning baseline (Section V-A.2) for a fair comparison.

2) Semi-Supervised Learning Baseline: For this baseline, the model is initially given the maximum allowed number of user-supplied labels (either 10% or 20%), and then uses self-training, which is a widely-used type of semi-supervised learning [4], [36] that iteratively assigns pseudo-labels to unlabelled samples. The system proceeds with pseudo-labelling the samples whose classification can be inferred with a confidence of > 0.9, choosing this high threshold to ensure that the system only pseudo-labels true positive and true negative samples. Despite leveraging both labelled and unlabelled data, the part of the data which the user is asked to initially label is arbitrarily chosen, so it is rarely the most diverse one. Additionally, users still need to manually set the model hyperparameters. To more strictly put SALTS to the test, we have chosen these hyperparameters carefully for the baseline, unlike how a non-ML expert would choose them in the real world. Specifically, the TensorFlow Keras model used in this semi-supervised baseline starts with a 1D convolutional layer, followed by batch normalisation and max pooling to extract hierarchical features and reduce dimensionality. A

convolutional layer is then applied, again followed by max pooling. The subsequent layer is an LSTM layer with 16 units, to capture long-range dependencies. Following this, a dense layer with a ReLU activation is included and the final output layer uses a softmax activation for classification. The model is trained using the Adam optimiser [32] and a focal loss function (see Section III-D), as this combination is the most suitable for integer-encoded labels.

3) AutoML Baseline: This baseline uses the Auto-SKLearn framework [46], an open-source and widely-used AutoML solution built on top of scikit-learn. We choose a randomly-picked (seed 111) 10% or 20% of the data that are manually labelled before feeding them to the system, and then rely on Auto-SKLearn for model development and training. This uses efficient Bayesian optimisation and automatically takes into consideration the performance of previous models on similar types of data for better results. It first initialises its Bayesian optimiser using meta-learning, and then evaluates a set of candidate models on the training dataset. Subsequently, it creates ensembles from these models and attempts to identify the optimal one.

4) Hyperparameter Tuning Baseline: In this baseline, we use a randomly-picked (seed 111) 10% or 20% of the data for manual labelling by the user, and then invoke Keras Tuner [47] to fit them to a neural network while optimising its hyperparameters. The model starts with a 1D convolutional layer with a dynamically-chosen activation function (eLU, ReLU, Leaky-ReLU, SeLU, or GeLU), and then uses a MaxPool1D layer, where the optimal pool size is automatically determined to enhance feature extraction capabilities. Another 1D convolutional layer follows this, maintaining flexibility in activation functions. Batch normalisation and max-pooling layers are incorporated for regularisation and downsampling with a dynamically-chosen pool size, and then another convolutional layer with similar configurations follows. The model ends with a dense layer being added and with its activation function (ReLU, Sigmoid, or TanH) subject to tuning. The optimisation then selects the most appropriate optimiser (Adam, AdaDelta, AdaGrad, AdaMax, or RmsProp) and the most appropriate value for the modulating factor γ of the focal loss function (see Section III-D). This baseline falls behind on the data input level as the labelled instances are arbitrarily chosen, and –similarly to AL– the model is not fitted to more samples than the user labels.

B. SALTS Algorithm Settings

The model architecture used in SALTS is the same as that of the hyperparameter tuning baseline of Section V-A.4, and the percentage of the data labelled per iteration is the same as in the AL baseline of Section V-A.1 for a fair comparison. The further settings we apply are described below.

Hyperparameter Optimisation Trials. For the data labelling phase of SALTS, we use the modAL framework [45]. Ahead of running it, we needed to determine the optimal number of hyperparameter optimisation trials per iteration of the joint data labelling and hyperparameter tuning phase (see Section III-B). We set these to < 3, informed by a comparative study using 20% of the ECG dataset's samples, where the best results were obtained with these values, achieving for instance an average accuracy of 0.969 and an average recall of 0.756. In our analysis, we tested the performance both with fewer (1) and more trials (4) per iteration. The experiment with fewer trials resulted in an accuracy of 0.961 and a recall value of 0.700, while the experiment with more trials achieved an accuracy of 0.966 and a recall of 0.773, indicating that additional trials beyond a certain point yield severely diminishing returns, if any, and just consume more computational resources.

TABLE I
RESULTS FROM THE EEG CLASSIFICATION EXPERIMENTS.

	Accuracy	Precision	Recall	F1 Score
	10% Labelling Budget			
Active Learning	0.882 ± 0.024	0.813 ± 0.026	0.915 ± 0.015	0.844 ± 0.027
Semi-Supervised	0.965 ± 0.001	0.971 ± 0.003	0.920 ± 0.006	0.943 ± 0.003
AutoML	0.960 ± 0.002	0.953 ± 0.005	0.919 ± 0.011	0.934 ± 0.003
HP Tuning	0.962 ± 0.006	0.963 ± 0.011	0.918 ± 0.019	0.938 ± 0.010
SALTS Proposal	**0.978 ± 0.003**	**0.976 ± 0.007**	**0.955 ± 0.011**	**0.965 ± 0.005**
	20% Labelling Budget			
Active Learning	0.893 ± 0.014	0.825 ± 0.015	0.921 ± 0.008	0.856 ± 0.015
Semi-Supervised	0.976 ± 0.001	0.978 ± 0.002	0.945 ± 0.004	0.961 ± 0.002
AutoML	0.965 ± 0.003	0.956 ± 0.001	0.933 ± 0.007	0.944 ± 0.004
HP Tuning	0.969 ± 0.007	0.975 ± 0.003	0.927 ± 0.021	0.948 ± 0.013
SALTS Proposal	**0.983 ± 0.003**	**0.984 ± 0.003**	**0.962 ± 0.009**	**0.972 ± 0.006**

Semi-Supervised Settings. For the automated training phase of SALTS that is inspired by semi-supervised learning, the algorithm iteratively pseudo-labels high-confidence samples with a confidence > 0.9, choosing the same threshold as the semi-supervised baseline towards a fair comparison. In our method, the model hyperparameters are tuned between each data acquisition step, to ensure that the process cooperates more effectively with the data labelling loop.

VI. EVALUATION RESULTS

All metrics for our experiments have been calculated on the test set and the results can be found in Tables I, II, and III. The study was approved by the ethics committee of the Department of Computer Science & Technology of the University of Cambridge (ethics review #2352).

EEG Classification. In EEG data, SALTS outperforms alternatives in all settings. In test accuracy, it reaches 0.978 when annotating 10% of the EEG samples, outperforming the semi-supervised (0.965), AutoML (0.960) and hyperparameter tuning (0.962) baselines, and achieving a nearly 10% increase in accuracy compared to the AL baseline (0.882). This trend continues to hold when annotating 20% of the EEG samples, as SALTS reaches an accuracy of 0.983, surpassing the semi-supervised (0.976), AutoML (0.965) and hyperparameter tuning (0.969) baselines, and achieving a 9% increase in accuracy compared to the AL baseline (0.893).

SALTS outperforms baselines in other widely-used metrics too, like the recall. With a 10% labelling budget, it reaches recall values beyond 0.95, while baselines range from 0.915 to 0.920. This indicates that it predicts correctly most of the relevant results, in contrast to the baselines. This trend

TABLE II

RESULTS FROM THE ECG CLASSIFICATION EXPERIMENTS.

	Accuracy	Precision	Recall	F1 Score
	10% Labelling Budget			
AutoDAL [16]	≈0.93	-	-	-
Active Learning	0.918 ± 0.009	0.559 ± 0.063	0.515 ± 0.016	0.516 ± 0.029
Semi-Supervised	0.930 ± 0.015	0.726 ± 0.030	0.508 ± 0.039	0.543 ± 0.040
AutoML	0.960 ± 0.002	**0.915 ± 0.011**	0.723 ± 0.022	**0.792 ± 0.018**
HP Tuning	0.903 ± 0.027	0.632 ± 0.117	0.473 ± 0.109	0.491 ± 0.078
SALTS Proposal	**0.962 ± 0.003**	0.885 ± 0.025	**0.732 ± 0.020**	0.770 ± 0.026
	20% Labelling Budget			
AutoDAL [16]	≈0.96	-	-	-
Active Learning	0.916 ± 0.020	0.553 ± 0.068	0.482 ± 0.067	0.495 ± 0.062
Semi-Supervised	0.950 ± 0.004	0.798 ± 0.093	0.586 ± 0.004	0.599 ± 0.018
AutoML	0.966 ± 0.000	0.924 ± 0.001	**0.764 ± 0.001**	**0.827 ± 0.001**
HP Tuning	0.903 ± 0.081	0.739 ± 0.057	0.629 ± 0.079	0.628 ± 0.135
SALTS Proposal	**0.969 ± 0.001**	**0.925 ± 0.021**	0.756 ± 0.007	0.796 ± 0.009

in the recall continues when experimenting with a 20% labelling budget too: SALTS reaches the best value of 0.962, outmatching all baselines. This demonstrates that SALTS uncovers information for EEG samples that each of the alternatives would not do alone given the same human effort.

ECG Classification. In ECG classification, SALTS reaches an accuracy of 0.962 with a 10% user-labelling limit and an accuracy of 0.969 with a 20% limit. For the 10% threshold, this shows that SALTS achieves at least a 6% increase in accuracy compared to the HP tuning baseline, and a 4% and 3% increase compared to the AL and semi-supervised baselines, respectively. Similarly, for the 20% threshold, SALTS achieves a 6% accuracy increase compared to the HP tuning baseline, a 5% increase compared to the AL baseline, and a 2% increase compared to the semi-supervised baseline. Of note, SALTS and the AutoML baseline perform similarly in accuracy, indicating that model parameters have the most impact for this data. In terms of the precision, SALTS outperforms all other methods when provided with a 20% labelling budget, reaching a value of 0.925.

In comparing SALTS with AutoDAL [16], when labelling a total of 10% of the ECG samples, we achieve an accuracy of ≈96%, compared to AutoDAL's ≈93%. This indicates that in settings where annotating unlabelled samples and adjusting model parameters is expensive, and medical experts can only spend fewer hours doing so, SALTS reaches superior accuracy with a single worker machine. When labelling 20% of the samples, we achieve an accuracy of ≈97%, compared to AutoDAL's ≈96%. Thus, we outperform the state-of-the-art, without the computational complexity of a distributed system. This is imperative when compute resources are more scarce, and more effectively reflects the real-world need of medical experts wanting to streamline ML model training.

IMU Classification. In HAR with IMU data, SALTS outperforms alternatives in all cases. It reaches an accuracy of 0.911 with a 10% data labelling limit, outperforming the hyperparameter tuning (0.799), AutoML (0.757) and semi-supervised (0.865) baselines, while reaching a significant 22% improvement with respect to the AL baseline (0.690). Similarly, for the 20% labelling threshold it reaches an accuracy of 0.925, thus exceeding the performance of the

TABLE III

RESULTS FROM THE IMU CLASSIFICATION EXPERIMENTS.

	Accuracy	Precision	Recall	F1 Score
	10% Labelling Budget			
Active Learning	0.690 ± 0.045	0.507 ± 0.025	0.532 ± 0.041	0.507 ± 0.035
Semi-Supervised	0.865 ± 0.012	0.814 ± 0.018	0.776 ± 0.012	0.771 ± 0.025
AutoML	0.757 ± 0.013	0.673 ± 0.048	0.626 ± 0.014	0.629 ± 0.022
HP Tuning	0.799 ± 0.028	0.685 ± 0.164	0.685 ± 0.068	0.654 ± 0.091
SALTS Proposal	**0.911 ± 0.017**	**0.884 ± 0.030**	**0.903 ± 0.016**	**0.884 ± 0.023**
	20% Labelling Budget			
Active Learning	0.722 ± 0.060	0.534 ± 0.110	0.581 ± 0.071	0.552 ± 0.088
Semi-Supervised	0.874 ± 0.016	0.858 ± 0.023	0.854 ± 0.013	0.842 ± 0.018
AutoML	0.821 ± 0.008	0.729 ± 0.032	0.705 ± 0.007	0.699 ± 0.010
HP Tuning	0.857 ± 0.030	0.801 ± 0.043	0.790 ± 0.062	0.790 ± 0.054
SALTS Proposal	**0.925 ± 0.015**	**0.895 ± 0.011**	**0.912 ± 0.013**	**0.898 ± 0.015**

hyperparameter tuning (0.857), the AutoML (0.821), the semi-supervised (0.874), and the AL (0.722) baselines.

In terms of the precision, recall and the F1 score, SALTS consistently reaches top values. Yet, the weaker performance of the baselines is also apparent in another, less quantitative area. The AL baseline requires too much effort from the user to tune its hyperparameters and is often less effective when not enough labels are supplied by the labellers, while absence of direct input from medical professionals in semi-supervised models also raises concerns about their suitability for real-world applications. The same applies for the AutoML and the hyperparameter tuning baselines, which cannot function on unlabelled data. This shows that each alternative approach is not sufficient alone, and highlights the importance of SALTS.

SALTS Performance. Across the examined case studies, SALTS performs best with a 20% labelling budget, dynamically choosing the queried data and hyperparameter combinations. For instance, in one of the trials for the ECG experiment with a 20% labelling budget, the system selects Leaky-ReLU amongst the possible activation functions for the 1D convolutional layers (see Sections V-A.4 and V-B), and 3 as the best pool size for the various MaxPool1D layers. Subsequently, it chooses ReLU as the best activation function for the penultimate dense layer, and continues with the tuning of 1.0 as the most effective γ value for the focal loss function, before finishing with the selection of Adam as the most appropriate optimiser. This dynamic approach, coupled with selecting informative samples, boosts SALTS' performance. The amount of labelled data required for SALTS to consistently outperform baselines is 10%, and our experiments indicate that its performance generally plateaus once 30% of the available samples have been annotated.

SALTS vs. the State-of-the-Art. As per Table II, SALTS outperforms AutoDAL (discussed in Section II) for the same amount of total labelled samples in the ECG data, achieving an increase of ≈3% in accuracy for a labelled data allowance of 10%, and an increase of ≈1% for an allowance of 20%. Similarly, it outperforms the state-of-the-art in AutoML too for most metrics and cases, achieving an increase of ≈2% in accuracy in the EEG dataset, as well as an increase of ≈15% for a labelled data allowance of 10% and an increase of ≈10% for an allowance of 20% in the IMU dataset.

SALTS Runtime. Unlike the baselines, which require manual intervention for tasks such as hyperparameter selection, SALTS automates these. Thus, comparing their runtime would overlook the additional human hours that the baselines require. Nevertheless, it is worth noting that SALTS needs ≈3 hours when operating on 20% of the samples of the ECG data. This scales approximately linearly with dataset size and sampling rates, and is based on testing performed on a hosted Jupyter notebook service that uses an NVIDIA GT 710, while having access to 12GB of its RAM. For context, the AL baseline takes ≈0.5 hour, the semi-supervised baseline ≈2.5 hours, the AutoML baseline ≈1 hour, and the HP tuning ≈2 hours. These come with an uncertainty of ±0.5 hour, and can fluctuate depending on the hardware.

VII. Conclusion

SALTS is a novel approach for practical learning on unlabelled healthcare time series, incorporating both model and human expertise to minimise user intervention. SALTS identifies the most informative samples and queries users to label them on the fly, while simultaneously automating a key part of model development through hyperparameter tuning. It proceeds to automatically infer the labels for high-confidence samples, letting the model be fitted to a significant amount of additional unlabelled samples with no further user input. We have demonstrated SALTS in both binary and multi-class classification of EEG, ECG, and IMU data, showcasing its applicability to a wide range of healthcare use cases.

ACKNOWLEDGMENT

This work is supported by Arm and by EPSRC grant EP/S023046/1 for the EPSRC Centre for Doctoral Training in Sensor Technologies and Applications.

References

[1] F. Peng, Q. Luo, and L. M. Ni, "ACTS: An Active Learning Method for Time Series Classification," *IEEE ICDE '17*, pp. 175–178, 2017.
[2] Y. Roh, G. Heo, and S. E. Whang, "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," *IEEE Trans. on Knowledge and Data Engineering*, vol. 33, no. 4, 2021.
[3] R. Adaimi and E. Thomaz, "Leveraging Active Learning and Conditional Mutual Information to Minimize Data Annotation in Human Activity Recognition," *ACM IMWUT*, vol. 3, no. 3, 2019.
[4] E. v. E. Jesper and H. H. Hoos, "A Survey on Semi-Supervised Learning," *Springer Machine Learning*, vol. 109, p. 373–440, 2019.
[5] C. A. Flores and R. Verschae, "A Generic Semi-Supervised and Active Learning Framework for Biomedical Text Classification," *EMBC '22*.
[6] B. Settles, "Active Learning Literature Survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.
[7] B. Settles and M. Craven, "An Analysis of Active Learning Strategies for Sequence Labeling Tasks," *EMNLP '08*, p. 1070–1079, 2008.
[8] L. Yang and A. Shami, "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice," *Neurocomputing*, 2020.
[9] S. Vavaroutas, L. Qendro, and C. Mascolo, "Uncertainty Estimation with Data Augmentation for Active Learning Tasks on Health Data," *IEEE EMBC '23*, 2023.
[10] C. I. Tang, I. Perez-Pozuelo, D. Spathis, S. Brage, N. Wareham, and C. Mascolo, "SelfHAR: Improving Human Activity Recognition through Self-training with Unlabeled Data," *ACM IMWUT*, 2021.
[11] J. Lim, J. Na, and N. Kwak, "Active Semi-Supervised Learning by Exploring Per-Sample Uncertainty and Consistency," 2023.
[12] M. Gao, Z. Zhang, G. Yu, S. Ö. Arik, L. S. Davis, and T. Pfister, "Consistency-Based Semi-Supervised Active Learning: Towards Minimizing Labeling Cost," *ECCV '20*, pp. 510–526, 2020.
[13] H. J. Escalante, "Automated Machine Learning: A Brief Review at the End of the Early Years," *CoRR*, vol. abs/2008.08516, 2020.
[14] A. Alsharef, K. Aggarwal, Sonia, M. Kumar, and A. Mishra, "Review of ML and AutoML Solutions to Forecast Time-Series Data," *Archives of Computational Methods in Engineering*, vol. 29, no. 7, 2022.
[15] J. Waring, C. Lindvall, and R. Umeton, "Automated Machine Learning: Review of the State-of-the-Art and Opportunities for Healthcare," *Artificial Intelligence in Medicine*, vol. 104, p. 101822, 2020.
[16] X. Chen and B. Wujek, "A Unified Framework for Automatic Distributed Active Learning," *IEEE Trans. on Pattern Analysis*, 2022.
[17] P. Koch, O. Golovidov, S. Gardner, B. Wujek, J. Griffin, and Y. Xu, "Autotune: A Derivative-Free Optimization Framework for Hyperparameter Tuning," *ACM KDD '18*, p. 443–452, 2018.
[18] J. D. Griffin and T. G. Kolda, "Nonlinearly-Constrained Optimization Using Heuristic Penalty Methods and Asynchronous Parallel Generating Set Search," *Applied Mathematics Research*, pp. 36–62, 2010.
[19] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," 2012.
[20] S. Roy, R. Mehera, R. K. Pal, and S. K. Bandyopadhyay, "Hyperparameter Optimization for Deep Neural Network Models: A Comprehensive Study on Methods and Techniques," *Springer*, 2023.
[21] P. Yue, Z. Li, M. Zhou, X. Wang, and P. Yang, "Wearable-Sensor-Based Weakly Supervised Parkinson's Disease Assessment with Data Augmentation," *Sensors*, vol. 24, no. 4, p. 1196, 2024.
[22] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid Training Data Creation with Weak Supervision," *Proc. of the VLDB Endowment*, vol. 11, no. 3, p. 269–282, 2017.
[23] I. Harmon, "A Brief Overview of Weak Supervision," 2020.
[24] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian Active Learning with Image Data," *ICML '17*, vol. 70, pp. 1183–1192, 2017.
[25] D. D. Lewis and J. Catlett, "Heterogeneous Uncertainty Sampling for Supervised Learning," *ML Proceedings '94*, pp. 148–156, 1994.
[26] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," 2016.
[27] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *CoRR*, vol. abs/1803.08375, 2018.
[28] A. L. Maas, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," *ICML '13*, 2013.
[29] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," *NeurIPS '17*, vol. 30, 2017.
[30] D. Hendrycks and K. Gimpel, "Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units," *CoRR*, 2016.
[31] S. Narayan, "The Generalized Sigmoid Activation Function: Competitive Supervised Learning," *Information Sciences*, vol. 99, no. 1, 1997.
[32] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014.
[33] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *CoRR*, vol. abs/1212.5701, 2012.
[34] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *JMLR*, vol. 12, 2011.
[35] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, "SemiBoost: Boosting for Semi-Supervised Learning," 2009.
[36] R. Dupre, J. Fajtl, V. Argyriou, and P. Remagnino, "Improving Dataset Volumes and Model Accuracy with Semi-Supervised Iterative Self-Learning," *IEEE Transactions on Image Processing*, vol. 29, 2020.
[37] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. S. Torr, and P. K. Dokania, "Calibrating Deep Neural Networks using Focal Loss," *NeurIPS '20*, vol. 33, pp. 15 288–15 299, 2020.
[38] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE ICCV '17*, pp. 2999–3007, 2017.
[39] A. Mavrin, "Focal Loss TensorFlow Implementation," 2019.
[40] Q. Wu and E. Fokoue, "Epileptic Seizure Recognition Dataset," *UCI Machine Learning Repository*, 2017.
[41] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "ECG Heartbeat Classification: A Deep Transferable Representation," *CoRR*, 2018.
[42] S. Fazeli, "ECG Heartbeat Categorization Dataset," *Kaggle*, 2018.
[43] G. B. Moody and R. G. Mark, "The Impact of the MIT-BIH Arrhythmia Database," *IEEE Engineering in Medicine and Biology*, 2001.
[44] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Mobile Sensor Data Anonymization," *IoTDI '19*, p. 49–58, 2019.
[45] T. Danka and P. Horváth, "modAL: A Modular Active Learning Framework for Python," *CoRR*, vol. abs/1805.00979, 2018.
[46] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Auto-sklearn: Efficient and Robust Automated Machine Learning," *Springer Automated Machine Learning*, pp. 113–134, 2019.
[47] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, and L. Invernizzi, "Keras Tuner," *Keras*, 2019.