SoQal: Selective Oracle Questioning for Consistency Based Active Learning of Cardiac Signals

Dani Kiyasseh [1]   **Tingting Zhu** [2]   **David Clifton** [2]

Abstract

Clinical settings are often characterized by abundant unlabelled data and limited labelled data. This is typically driven by the high burden placed on oracles (e.g., physicians) to provide annotations. One way to mitigate this burden is via active learning (AL) which involves the (a) acquisition and (b) annotation of informative unlabelled instances. Whereas previous work addresses either one of these elements independently, we propose an AL framework that addresses both. For acquisition, we propose Bayesian Active Learning by Consistency (BALC), a sub-framework which perturbs both instances and network parameters and quantifies changes in the network output probability distribution. For annotation, we propose SoQal, a sub-framework that dynamically determines whether, for each acquired unlabelled instance, to request a label from an oracle or to pseudo-label it instead. We show that BALC can outperform start-of-the-art acquisition functions such as BALD, and SoQal outperforms baseline methods even in the presence of a noisy oracle.

1. Introduction

Deep learning algorithms often need access to abundant, high-quality, labelled data. Such access, though, is a rarity within healthcare. For example, in low-resource clinical settings, data can be collected in abundance via wearable sensors however physicians, who are typically required to annotate such data, are either in low-supply or ill-trained to complete the task. As such, these settings are often characterized by abundant *unlabelled* data and limited *labelled* data. In light of this observation, we focus on the following question: *how can we design clinical algorithms that exploit abundant, unlabelled data and limited, labelled data while minimizing the labelling burden placed on physicians?*

One way to address this question is via the active learning (AL) framework (Settles, 2009) which iterates over three main steps. 1) *acquisition* - a learner (e.g., neural network) is tasked with acquiring unlabelled instances 2) *annotation* - an oracle (e.g., physician) is tasked with labelling such acquired instances, and 3) *aggregation* - the learner is trained on the existing and newly-labelled instances. Whereas previous work addresses either one of the first two steps of active learning, in this work, we aim to modify both.

The acquisition of unlabelled instances is commonly performed via an acquisition function, an example of which is Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011). It quantifies the degree to which a network is uncertain about the classification of an instance. To do so, Monte Carlo Dropout (MCD) (Gal & Ghahramani, 2016) is exploited wherein network parameters are stochastically perturbed while an unlabelled instance is passed through a network. These parameter perturbations manifest in the form of distinct hypotheses (decision boundaries), as shown in Fig. 1 (left). However, MCD, as we later outline, can erroneously overlook informative instances for acquisition due to its misspecification of parameter perturbations. As for annotating acquired instances, it is often assumed that oracles are continuously available and sufficiently skilled to provide such annotations, an assumption that is rarely satisfied within healthcare settings.

In this paper, we make the following contributions. **First**, we propose an AL framework, Monte Carlo Perturbations (MCP), that involves perturbing instances and observing concomitant changes in the network output distribution. We show that MCP performs on par with MCD in several settings. **Second**, we take inspiration from consistency training and propose an AL framework, Bayesian Active Learning by Consistency (BALC), that involves perturbing *both* instances and parameters and observing changes in the output distribution. Moving away from uncertainty-based acquisition functions, we introduce two *consistency-based* acquisition functions, $BALC_{KLD}$ and $BALC_{JSD}$, and show that they can outperform state-of-the-art acquisition functions

---

[1]Department of Computing and Mathematical Sciences, California Institute of Technology, California, USA [2]Department of Engineering Science, University of Oxford, Oxford, UK. Correspondence to: Dani Kiyasseh <dkiyass1@caltech.edu>.
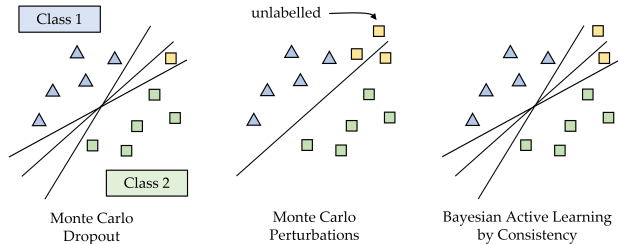
Figure 1: **Perturbation frameworks used in active learning.** Labelled instances from two classes and unlabelled instances (yellow) with the hypotheses of **(Left)** Monte Carlo Dropout (MCD), where each MC sample is a distinct hypothesis, **(Centre)** Monte Carlo Perturbations (MCP), with one hypothesis but several perturbed variants of the unlabelled instance, and **(Right)** Bayesian Active Learning by Consistency (BALC) with multiple hypotheses and perturbed instances. In this paper, we introduce MCP and BALC as alternatives to MCD in order to acquire unlabelled instances.

such as BALD. **Third**, existing acquisition functions are static; they determine the informativeness of an instance at a single snapshot in time. Instead, we propose the simple modification of *tracking* the value of an acquisition function over time (epochs). We show that acquisition functions with such temporal information can outperform their static counterparts. **Fourth**, we take inspiration from selective classification and propose SoQal, a framework which determines whether, for each acquired unlabelled instance, to request a label from an oracle or to pseudo-label it instead. We show that SoQal outperforms state-of-the-art selective classification methods. To the best of our knowledge, we are the first to explore these avenues in the context of cardiac signals.

2. Related Work

2.1. Active Learning in Healthcare

For medical images, previous work has acquired instances by measuring their distance in a latent space to images in the training set (Smailagic et al., 2018; 2020). For time-series data, researchers have acquired instances from electronic health record (EHR) (Gong et al., 2019) and electrocardiogram (ECG) (Wiens & Guttag, 2010; Pasolli & Melgani, 2010; Wang et al., 2019; Kiyasseh et al., 2021a) databases. For example, although Kiyasseh et al. (2021a) acquire cardiac signals, they do so in the context of continual learning and do not explore how to annotate such cardiac signals. More generally, Gal et al. (2017) adopt BALD (Houlsby et al., 2011) in the context of Monte Carlo Dropout to acquire instances that maximize the Jensen-Shannon divergence (JSD) across MC samples. For an in-depth review of

active learning, we refer readers to Settles (2009).

2.2. Oracles in Active Learning

Previous work attempts to learn from multiple or imperfect oracles (Dekel et al., 2012; Zhang & Chaudhuri, 2015; Sinha et al., 2019). For example, Urner *et al.* (Urner et al., 2012) identify a suitable oracle to label a particular instance. In contrast to our work, they do not explore the independence of a learner from an oracle. Although Yan et al. (2016) consider oracle abstention, we place the decision of abstention under the control of the learner. To the best of our knowledge, we are the first to explore a dynamic oracle selection strategy in the context of cardiac signals.

2.3. Consistency Training

Consistency training helps enforce the smoothness assumption (McCallumzy & Nigamy, 1998; Zhu, 2005; Verma et al., 2019). For example, Xie et al. (2019) penalize networks for generating drastically different outputs in response to perturbed instances, resulting in perturbation-invariant representations. Most similar to our work is that of Gao et al. (2020) which designs a consistency-loss based on the Kullback-Leibler divergence $\mathcal{D}_{KL}$. Whereas they actively acquire instances using the variance of the probability assigned to each class by the network in response to perturbed versions of the same instance, we design distinct consistency-based acquisition functions.

2.4. Selective Classification

Selective classification imbues a network with the ability to abstain from making predictions (Chow, 1970; El-Yaniv & Wiener, 2010; Cortes et al., 2016). Wiener & El-Yaniv (2011) use a support vector machine to rank and reject instances based on the degree of disagreement between hypotheses. In some frameworks, these are the same instances that active learning views as most informative. Although Liu et al. (2019) propose the gambler's loss to learn a selection function that determines whether instances are rejected, this approach is not implemented in the context of active learning. SelectiveNet (Geifman & El-Yaniv, 2019) introduces a multi-head architecture similar to ours, however it assumes the presence of ground-truth labels and, therefore, does not trivially extend to unlabelled instances.

3. Background

3.1. Active Learning

Consider a learner $f_\theta : \boldsymbol{x} \in \mathbb{R}^D \rightarrow \boldsymbol{v} \in \mathbb{R}^E$, parameterized by $\boldsymbol{\theta}$, that maps a D-dimensional instance, $\boldsymbol{x}$, to an E-dimensional representation, $\boldsymbol{v}$. Further consider a function, $p_\omega : \boldsymbol{v} \in \mathbb{R}^E \rightarrow y \in \mathbb{R}^C$, that maps an E-dimensional

representation, $\boldsymbol{v}$, to a C-dimensional output, y, where C is the number of classes. After training on a pool of labelled data, $\mathcal{D}_L = \{\boldsymbol{X_L}, \boldsymbol{Y_L}\}$ for τ epochs, the learner is tasked with querying the unlabelled pool of data, $\boldsymbol{X_U}$, and acquiring the top b fraction of instances, $\{\boldsymbol{x}_i\}_{i=1}^b \sim \boldsymbol{X_U}$, that it deems to be most informative. The degree of informativeness of an instance, $\boldsymbol{x}$, is determined by an acquisition function, $\alpha(\boldsymbol{x}) \in \mathbb{R}$, such as Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011). These functions are typically used alongside Monte Carlo Dropout (MCD) (Gal & Ghahramani, 2016) to identify instances that lie in the region of uncertainty, a region in which hypotheses disagree the most about instances (Fig. 1 left).

3.2. Consistency Training

Consider an unlabelled instance, $\boldsymbol{x} \sim \boldsymbol{X_U}$, and its perturbed counterpart, $\boldsymbol{x'} = \boldsymbol{x} + \epsilon$, where ϵ is some perturbation. A network is said to be invariant to such a perturbation if its outputs, $p_\omega(\boldsymbol{x})$ and $p_\omega(\boldsymbol{x'})$, are similar to one another. Consistency training is one way to encourage such an invariance. In this work, we exploit this intuition to design acquisition functions, as outlined next.

4. Methods

4.1. Monte Carlo Perturbations

Acquisition functions dependent upon parameter perturbations, such as in MCD, can overlook, and thus fail to acquire, informative unlabelled instances. To see this, and without loss of generality, consider an unlabelled instance which is (a) in proximity to a decision boundary, thus deeming it informative for training (Settles, 2009) and (b) classified by a network into an arbitrary class. In this setting, a total of T parameter perturbations results in T hypotheses, altering the outputs of a network, $\{p_\omega^i\}_{i=1}^T$, in response to an unlabelled instance. We visualize the distribution of such outputs in Fig. 2 (red rectangle), after having applied $T = 3$ parameter perturbations. If the perturbations happen to be too small in magnitude, for example, then a network will exhibit a similar output distribution ($p_\omega^1 = p_\omega^2 = \cdots$) across the parameter perturbations. However, since acquisition functions assign value based on *changes* in the output distribution, this informative unlabelled instance (due to its proximity to decision boundary) would be *erroneously* deemed uninformative.

One way to avoid missing these informative instances is by stochastically perturbing instances (instead of network parameters) and observing changes in the network outputs, a setup we refer to as Monte Carlo Perturbations (MCP). This results in a single hypothesis but multiple perturbed variants of the instance (see Fig. 1 centre). The intuition is that network outputs will differ more significantly for an
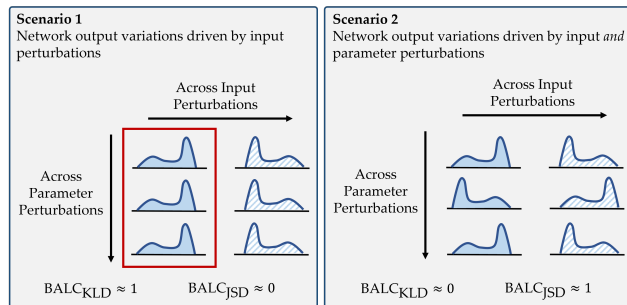

Figure 2: **Scenarios demonstrating the effect of input and parameter perturbations on the behaviour of the proposed acquisition functions,** BALC_{KLD} **and** BALC_{JSD}**. (Scenario 1)** network output variations caused primarily by input perturbations. The red rectangle illustrates how parameter perturbations alone, as with MCD, can be insufficient in causing changes to the network outputs, thus failing to acquire informative unlabelled instances. **(Scenario 2)** network output variations caused by both input and parameter perturbations. We show that while BALC_{KLD} is likely to acquire instances due to input perturbations, BALC_{JSD} considers both input and parameter perturbations when performing acquisitions.

instance in proximity to the decision boundary than for an instance farther away. By quantifying these output changes, as is done with almost any acquisition function, we can identify informative instances for acquisition. The main advantage of MCP over MCD is the increased control and interpretability of the applied perturbations; perturbations applied to instances are likely to be more understandable than those applied to parameters. A formal derivation of MCP can be found Appendix B.

4.2. Bayesian Active Learning by Consistency

It could be argued that the same limitations exhibited by MCD also extend to MCP. After all, both frameworks apply perturbations to either network parameters or instances. Acknowledging this, we propose a framework, entitled Bayesian Active Learning by Consistency (BALC), in which perturbations are simultaneously applied to network parameters *and* instances. As such, this results in multiple decision boundaries and perturbed instances (see Fig. 1 right). BALC consists of three main steps: 1) perturb an instance, $\boldsymbol{x} \in \mathbb{R}^D$, to generate $\boldsymbol{x'} \in \mathbb{R}^D$, 2) perturb the network parameters, $\boldsymbol{\theta}$, to generate $\boldsymbol{\theta'}$, and 3) pass both instances, $\boldsymbol{x}$ and $\boldsymbol{x'}$, through the *perturbed* network, generating outputs, $p(y|\boldsymbol{x}, \boldsymbol{\theta'})$ and $p(y|\boldsymbol{x'}, \boldsymbol{\theta'}) \in \mathbb{R}^C$, respectively. Note that we drop the explicit dependence on $\boldsymbol{\omega}$ for clarity. We perform these steps for T stochastic parameter perturbations and generate two matrices of network outputs, $\boldsymbol{G}(\boldsymbol{x}), \boldsymbol{G'}(\boldsymbol{x'}) \in \mathbb{R}^{T \times C}$, as shown in Fig. 3.
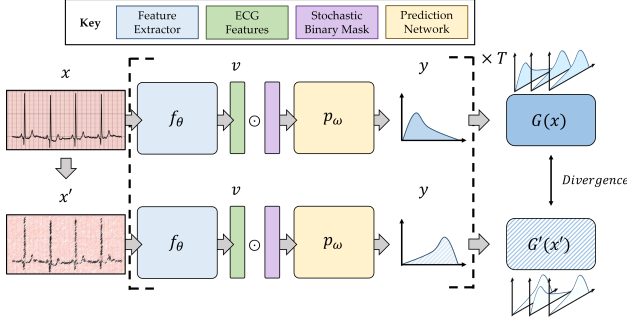
Figure 3: **Consistency-based active-learning framework.** We perturb an instance, $\boldsymbol{x}$, to generate, $\boldsymbol{x}'$, and extract their corresponding representations, v. We apply a stochastic dropout mask to these representations and obtain an output distribution over the classes. When repeated T times, this generates a pair of matrices, $\boldsymbol{G}(\boldsymbol{x})$ and $\boldsymbol{G}'(\boldsymbol{x}')$, whose divergence is calculated via BALC_{KLD} or BALC_{JSD}.

The intuition is that the greater the divergence between $\boldsymbol{G}$ and $\boldsymbol{G}'$, the closer an instance is to the decision boundary. To that end, we propose two consistency-based acquisition functions, BALC_{KLD} and BALC_{JSD}.

To calculate $\text{BALC}_{\text{KLD}}(\boldsymbol{x}) \in \mathbb{R}$ (1), we first empirically fit two C-dimensional Gaussian distributions, $\mathcal{N}(\boldsymbol{x})$ and $\mathcal{N}(\boldsymbol{x}')$ to $\boldsymbol{G}$ and $\boldsymbol{G}'$, respectively. For each such matrix, we obtain an empirical mean vector, $\boldsymbol{\mu} = \frac{1}{T}\sum_i^T \boldsymbol{G}_i$ and covariance matrix, $\boldsymbol{\Sigma} = (\boldsymbol{G} - \boldsymbol{\mu})^T(\boldsymbol{G} - \boldsymbol{\mu})$. We then quantify the D_{KL} between $\mathcal{N}(\boldsymbol{x})$ and $\mathcal{N}(\boldsymbol{x}')$ as follows.

$$\text{BALC}_{\text{KLD}}(\boldsymbol{x}) = \mathcal{D}_{KL}(\mathcal{N}(\boldsymbol{x}) \parallel \mathcal{N}(\boldsymbol{x}')) \qquad (1)$$

where $\mathcal{N}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x}))$ and $\mathcal{N}(\boldsymbol{x}') = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}'), \boldsymbol{\Sigma}(\boldsymbol{x}'))$.

We note that BALC_{KLD} is likely to detect changes in the network outputs due solely to instance perturbations. To see this, consider Scenario 1 and 2 presented in Fig. 2. Changes in network outputs are caused by either instance perturbations alone (Scenario 1) or both instance and parameter perturbations (Scenario 2). In these two scenarios, $\text{BALC}_{\text{KLD}} \approx 1$ and 0, respectively. Since the informativeness of an instance is related to the magnitude of the acquisition function, these scenarios suggest that BALC_{KLD} has a preference for instance perturbations. In order to detect changes in the network outputs due to *both* instance and parameter perturbations, we introduce BALC_{JSD}. Support for this claim can also be found in Fig. 2.

$\text{BALC}_{\text{JSD}}(\boldsymbol{x}) \in \mathbb{R}$ (2) comprises the difference of two terms. The first term, Ⓐ, calculates the D_{KL} of network outputs due to a single instance perturbation and averages this across T parameter perturbations. The second term, Ⓑ, averages the network outputs across parameter perturbations independently for the original and perturbed instance before

calculating the D_{KL} of the resulting mean outputs. The full derivation of BALC_{JSD} can be found in Appendix C.

$$\text{BALC}_{\text{JSD}}(\boldsymbol{x}) = \overbrace{\frac{1}{T}\sum_{i=1}^{T}\left[\mathcal{D}_{KL}(\boldsymbol{G}_i(\boldsymbol{x}) \parallel \boldsymbol{G}'_i(\boldsymbol{x}'))\right]}^{\text{Ⓐ across parameter perturbations}} - $$

$$\underbrace{\mathcal{D}_{KL}(\frac{1}{T}\sum_{i=1}^{T}\boldsymbol{G}_i(\boldsymbol{x}) \parallel \frac{1}{T}\sum_{i=1}^{T}\boldsymbol{G}'_i(\boldsymbol{x}'))}_{\text{Ⓑ across input perturbations}} \qquad (2)$$

4.3. Tracked Acquisition Function

So far, we have discussed active learning frameworks that, similar to those in the literature, quantify the informativeness of an unlabelled instance at a *single* snapshot in time (e.g., at a particular epoch, τ). This static setup, however, faces two limitations. First, it depends on an appropriate choice of epoch for the acquisition of instances, which is non-trivial to identify a priori. For example, an acquisition function can be of little value if calculated when network parameters have yet to be updated sufficiently. Second, the diversity and number of hypotheses obtained via parameter perturbations can be limited by this single-epoch view. This is detrimental given the established benefit of eliminating unsuitable hypotheses at a greater rate ($\downarrow$ version space) (Cohn et al., 1994).

To overcome such limitations, we propose to *track* this acquisition function over time (e.g., epochs) before deploying it for the acquisition of instances. This is a simple extension to almost any acquisition function. In the process, we become less dependent on the choice of acquisition epoch and are likely to increase the diversity of hypotheses at our disposal. Formally, consider an acquisition function, $\alpha(\boldsymbol{x})$, which would ordinarily be calculated once at epoch, τ, for each instance, $\boldsymbol{x}$. In our formulation, we calculate $\alpha(\boldsymbol{x}, t)$ at each epoch, $t \in [1, \tau]$. At epoch, τ, which we refer to as an *acquisition epoch*, the modified acquisition function, $\text{AUTAF}(\boldsymbol{x}, \alpha) \in \mathbb{R}$, corresponds to the area under the tracked acquisition function, approximated via the trapezoidal rule.

$$\text{AUTAF}(\boldsymbol{x}, \alpha) \approx \sum_{t=1}^{\tau}\left(\frac{\alpha(\boldsymbol{x}, t + \Delta t) + \alpha(\boldsymbol{x}, t)}{2}\right)\Delta t \quad (3)$$

where Δt is the interval between epochs at which acquisition values are calculated.

4.4. Selective Oracle Questioning

Up until this point, we have discussed methods that exclusively target the *acquisition* of unlabelled instances. We now direct our attention to the *annotation* of such instances with

the goal of minimizing the labelling burden that is placed on an oracle. To do so, we design a framework that, given an acquired unlabelled instance, dynamically determines whether to request a label from an oracle or to pseudo-label that instance instead. This framework, which we refer to as selective oracle questioning in active learning, SoQal, is outlined next.

Oracle selection network   Let us first consider a learner, $g_\phi : v \in \mathbb{R}^E \to r \in [0, 1]$, parameterized by ϕ, that maps an E-dimensional representation, v, to a scalar, r, as shown in Fig. 4. We refer to this learner as an oracle selection network.

Learning a proxy for misclassifications   The intuition underlying our oracle selection framework is as follows. Given an instance, a network should request a label (i.e., ask for help) from an oracle instead of pseudo-label it if the network is likely to misclassify this instance. This idea operates under the assumption that the network is able to identify whether an instance will be misclassified. While this is possible with *labelled* data, it is non-trivial with *unlabelled* data (due to absence of a ground-truth), the prime focus of active learning. As such, we need a way to identify whether an unlabelled instance would have been misclassified if it were to be pseudo-labelled by the network. In other words, we need a reliable proxy for a misclassification.

To design this proxy, we exploit the oracle selection network (g_ϕ) and its scalar output, r. As we explain next, we learn ϕ such that higher values of r correspond to misclassifications. Achieving this requires that r have a supervisory ground-truth label. We choose such a label to be the zero-one loss, $e \in \mathbb{R}$, incurred by the prediction network, p_ω, on an instance, x. Intuitively, when $e = 1$, the network should request a label from an oracle. Note that this ground-truth label, e, is only available while training on *labelled* data. We later explain how to exploit r on acquired *unla-*
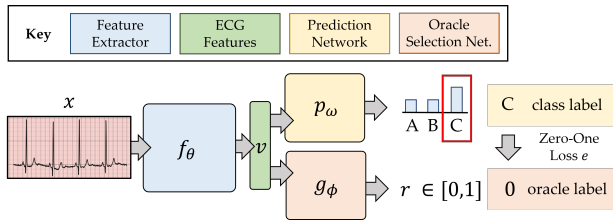

Figure 4: **Selective oracle questioning framework.** Representation, v, of instance, x is passed through p_ω to generate task predictions and g_ϕ to determine whether a label is requested from an oracle. The zero-one loss of p_ω is the ground-truth label for r. High (low) values of r indicate that an instance should be labelled by an oracle (pseudo-labelled by prediction network).

belled instances. As such, while training on labelled data, we optimize (4) with a categorical cross-entropy loss for the prediction network, $\mathcal{L}_{CPL}$, with ground-truth class labels, c, and a weighted binary cross-entropy loss for the oracle selection network, $\mathcal{L}_{OSL}$. In the process, we learn the parameters, θ, ω, and ϕ in an end-to-end manner.

$$\mathcal{L} = \mathcal{L}_{CPL} + \mathcal{L}_{OSL} \qquad (4)$$

$$\mathcal{L}_{CPL} = -\sum_{i=1}^{B} \log\left(p_\omega(y_i = c | x_i, \phi)\right)$$

$$\mathcal{L}_{OSL} = -\sum_{i=1}^{B} \beta e_i \log g_\phi(x_i) - (1 - e_i) \log\left(1 - g_\phi(x_i)\right)$$

Weighted oracle selection loss. During the early stages of training on *labelled* data, a network struggles to classify instances correctly. In our framework, this implies that we will encounter more terms with $e = 1$ (misclassification) than those with $e = 0$ (correct classification). The opposite is true as training progresses and the network becomes more adept at classifying instances. Therefore, regardless of the stage of training (early vs. late), there will be an imbalance in the ground-truth labels, e, provided to the oracle selection network. This imbalance sends a strong supervisory signal to g_ϕ through the oracle selection loss, $\mathcal{L}_{OSL}$, and not accounting for it would lead to systematically higher (lower) r values during the early (late) stages of training. This can reduce the reliability of r as a proxy for misclassifications. As such, we introduce a dynamic weight, $\beta = \frac{\sum \mathbb{1}(e=0)}{\sum \mathbb{1}(e=1)}$, where $\mathbb{1}$ is the indicator function. As training progress, $\beta < 1 \to \beta > 1$, as the ratio of correctly classified ($e = 0$) to misclassified ($e = 1$) instances within a mini-batch changes.

Making decisions with proxy   We exploit r for the binary decision of whether to request a label from an oracle or to generate a pseudo-label instead. Since $r \in [0, 1]$, one simple rule would be to select an oracle if $r > 0.5$. This threshold, however, may be sub-optimal particularly if the values of r are not calibrated. Therefore, to design a robust rule, we account for the *distribution* of the r values stratified according to the zero-one loss, e, on *labelled* data, and the separability of such distributions. We present such empirical stratified distributions in Figs. 5a and 5b during the early and late stages of training, respectively. As expected, we find that correctly-classified instances ($e = 0$) tend to have lower r values than those which are misclassified.

We now outline the steps involved in making the binary decision. First, after each training epoch, t, we fit two unimodal Gaussian distributions, $\mathcal{N}_0(\mu_0, \sigma_0^2)$ and $\mathcal{N}_1(\mu_1, \sigma_1^2)$ corresponding to $e = 0$ and $e = 1$, respectively, to the r values (generated on the *labelled* data), similar to those in Fig. 5b.
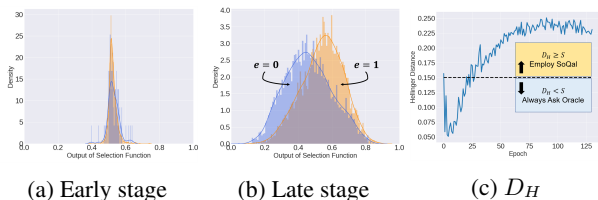
(a) Early stage   (b) Late stage   (c) D_H

Figure 5: **Components of selective oracle questioning framework.** Distribution of the outputs, r, of g_ϕ stratified based on the zero-one loss, e during the (a) early and (b) late stages of training. (c) Hellinger distance, D_H, between the two distributions of r increases during training.

Low separability of such distributions would suggest that the oracle selection network has yet to learn to distinguish between correctly-classified and misclassified instances, and is thus generating an unreliable proxy, r. We quantify this separability via the Hellinger distance, $D_H \in [0, 1]$, a metric to which a threshold can be easily applied. In Fig. 5c, we show that indeed as training progresses, the separability of the two distributions increases, which is suggestive of an increasingly reliable proxy. Based on some user-defined threshold, S, low separability (i.e., low proxy reliability) can be defined as $D_H < S$. In such an event, we use a conservative strategy that defers labelling to the oracle. This value of S can be altered depending on the relative level of trust placed in the network and oracle.

Pseudo-labelling. High separability, defined as $D_H \geq S$ suggests that the proxy is sufficiently reliable to make decisions. As such, for each acquired *unlabelled* instance, $\boldsymbol{x}_u \sim \boldsymbol{X_U}$, we obtain $r_u = g_\phi(\boldsymbol{x}_u)$ and compare the values of $\mathcal{N}_0$ and $\mathcal{N}_1$ when evaluated at r_u. If $\mathcal{N}_1 > \mathcal{N}_0$, then the value of r is too high (indicating a potential network misclassification), and a label is requested from an oracle. Otherwise, the instance is pseudo-labelled via $\arg\max_c p_\omega(y = c | \boldsymbol{x})$. The probability of requesting a label from an oracle is denoted by $p(\mathrm{A})$.

$$p(\mathrm{A}) = \begin{cases} 1, & D_H < S \\ 1, & \mathcal{N}_1 > \mathcal{N}_0 \text{ and } D_H \geq S \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $\mathcal{N}_1 = \mathcal{N}\left(r = r_u | \mu_1, \sigma_1^2, e = 1\right)$ and $\mathcal{N}_0 = \mathcal{N}\left(r = r_u | \mu_0, \sigma_0^2, e = 0\right)$. We note that our oracle selection framework is independent of the acquisition of unlabelled instances, making it general enough to be used alongside almost any acquisition function. Pseudo-code for the entire pipeline can be found in Appendix F.

5. Experimental Design

5.1. Datasets

We conduct our experiments[1] using PyTorch (Paszke et al., 2019) on four publicly-available datasets comprising cardiac signals, such as the photoplethysmogram (PPG) and the electrocardiogram (ECG) alongside cardiac arrhythmia labels (abnormalities in the functioning of the heart). **PhysioNet 2015** ($\mathcal{D}_1$) (Clifford et al., 2015) consists of PPG data alongside 5 different classes of cardiac arrhythmia. $\mathcal{D}_2$ is similar to the first however comprises ECG data. **PhysioNet 2017** ($\mathcal{D}_3$) (Clifford et al., 2017) consists of 8,528 single-lead ECG recordings alongside 4 different classes of cardiac arrhythmia. **Cardiology** ($\mathcal{D}_4$) (Hannun et al., 2019) consists of single-lead ECG data from 292 patients alongside 12 different classes of cardiac arrhythmia.

In order to evaluate our framework in the limited data regime, we place a fraction, $F \in [0.1, 0.3, 0.5, 0.7, 0.9]$, of the training dataset into a labelled set, $\mathcal{D}_L$. and its complement into an unlabelled set, $\boldsymbol{X_U}$ (see Table 1). Additional details about the datasets and pre-processing steps can be found in Appendix D.

Dataset	Train		Val.	Test
	Labelled	Unlabelled		
$\mathcal{D}_1$	401	4233	1124	1435
$\mathcal{D}_2$	401	4233	1124	1435
$\mathcal{D}_3$	1,776	16,479	4,582	5,824
$\mathcal{D}_4$	452	4,110	1,131	1,386

Table 1: **Number of instances in the training, validation, and test sets.** The distribution of labelled instances is shown for a fraction, $F = 0.1$, of the original training set. This is also used for the selective oracle questioning experiments. The remaining distributions can be found in Appendix D.

5.2. Network Architecture

Our network architecture involves a lightweight convolutional network which receives a cardiac time-series segment (2500 samples or ≈ 5 seconds in duration) as input and returns a probability distribution over cardiac arrhythmias as output. We chose this architecture based on previous research demonstrating its effectiveness in classifying cardiac arrhythmias (Kiyasseh et al., 2021b). Additional details about the network architecture can be found in Appendix E.

5.3. Active Learning Scenarios

We explore three distinct active learning scenarios characterized by the presence and quality of an oracle. Recall that the motivation behind our framework was to exploit abundant

---

[1]Code: https://github.com/danikiyasseh/SoQal

unlabelled and limited labelled data while alleviating (and not necessarily eliminating) the labelling burden placed on an oracle. As such, the target and realistic clinical scenario in which our framework would be deployed is one where an oracle (e.g., physician) is available in some capacity. However, to explore the extreme limits of our approach, and as a stepping stone to the target clinical scenario, we begin our experimentation without an oracle and transition to the scenarios in which an oracle is available.

Scenario 1 - Without Oracle. we assume that a physician is unavailable to provide labels and thus evaluate the performance of our framework *without* an oracle.

Scenario 2 - Noise-free Oracle. we assume that a physician is available and capable of providing accurate labels.

Scenario 3 - Noisy Oracle. we assume that a physician is either ill-trained or unable to perform a diagnosis due to its difficulty. To simulate this setting, we introduce two types of label noise. We stochastically flip the ground-truth label (unseen by the network) of each unlabelled instance to 1) **(Random)** any other label randomly, or 2) **(Nearest Neighbour)** its nearest neighbour, in a smaller dimensional subspace, from a *different* class. Whereas the first form of noise is extreme, the latter is more realistic as it may reflect uncertain physician diagnoses. Furthermore, we simulate noise of different magnitude by injecting it with probability $\gamma = [0.05, 0.1, 0.2, 0.4, 0.8]$.

5.4. Baselines

We compare our proposed acquisition functions to the state-of-the-art functions used alongside MCD. These include **Var Ratio**, **Entropy**, and **BALD** (Houlsby et al., 2011), definitions of which can be found in Appendix A. We also experiment with several baselines that exhibit varying degrees of oracle dependence. ϵ**-greedy** (Watkins, 1989) - a stochastic strategy that we adapt to exponentially decay the reliance of the network on an oracle as a function of the number of acquisition epochs. S**-response** - assumes that high entropy predictions are indicative of instances that the network is unsure of. Therefore, we introduce a threshold, S_{Entropy}, such that if it is exceeded, an oracle is requested to label the chosen instance (see Appendix E).

5.5. Hyperparameters

For all experiments, we chose the number of MC samples $T = 20$ to balance between computational complexity and accuracy of the approximation of the version space. During training, we acquire unlabelled instances at pre-defined epochs (acquisition epochs), $\tau = 5n, n \in \mathbb{N}^+$. During each acquisition epoch, we acquire $b = 2\%$ of the remaining unlabelled instances. We investigate the effect of such hyperparameters on performance in Appendices K-M. When exper-

imenting with tracked acquisition functions, we chose the temporal period, $\Delta t = 1$, calculating the acquisition function at each epoch of training. Given the increasing trend of D_H during training (see Fig. 5c), we chose $D_H \geq S = 0.15$ to balance between the reliability of the proxy and the independence of the network from an oracle.

6. Experimental Results

6.1. Active Learning without Oracle

We begin by exploring the performance of active learning frameworks without an oracle. In Fig. 6a, we illustrate the validation AUC of a network that is initially exposed to $F = 0.3$ of the labelled training data of $\mathcal{D}_2$.

We find that a network which exploits BALC_{KLD} learns faster than, and outperforms, those which exploit the remaining acquisition functions. For example, BALC_{KLD} and BALD_{MCD} achieve AUC ≈ 0.69 after 20 and 40 epochs of training, respectively, reflecting a two-fold increase in learning efficiency. Moreover, BALC_{KLD} and $\text{Entropy}_{\text{MCD}}$ achieve a final AUC ≈ 0.72 and ≈ 0.70, respectively. One hypothesis for this improved performance is that BALC_{KLD}, as a consistency-based active learning framework, acquires unlabelled instances which may still be correctly pseudo-labelled by the network despite the absence of an oracle. This, in turn, facilitates learning. Uncertainty-based acquisition functions, on the other hand, acquire unlabelled instances to which the network is most uncertain. As such, pseudo-labels for these instances are likely to be incorrect. The results for the remaining experiments can be found in Appendix G.

We now transition to quantifying the marginal benefit of incorporating temporal information into the acquisition functions. In Fig. 6b, the panel on the left illustrates the percent change in the AUC when comparing MCP to MCD while using *static* acquisition functions. The panel on the right, however, depicts the results after having incorporated temporal information into the acquisition functions used alongside MCP. Hence, the naming MCP Temporal. We find that tracked acquisition functions add value when the size of the labelled dataset is small ($\downarrow F$) (red rectangle). For example, transitioning from MCP Static to MCP Temporal when exploiting BALD at $F = 0.1$ improves the AUC by 11%. Similar improvements can be seen in Appendix H. We hypothesize that this improvement is due to the increased diversity, and number, of hypotheses available when considering temporal information, thus eliminating unsuitable hypotheses at a greater rate.

6.2. Active Learning with Noise-free Oracle

Having explored the performance of active learning frameworks without an oracle, we now assume the presence of a
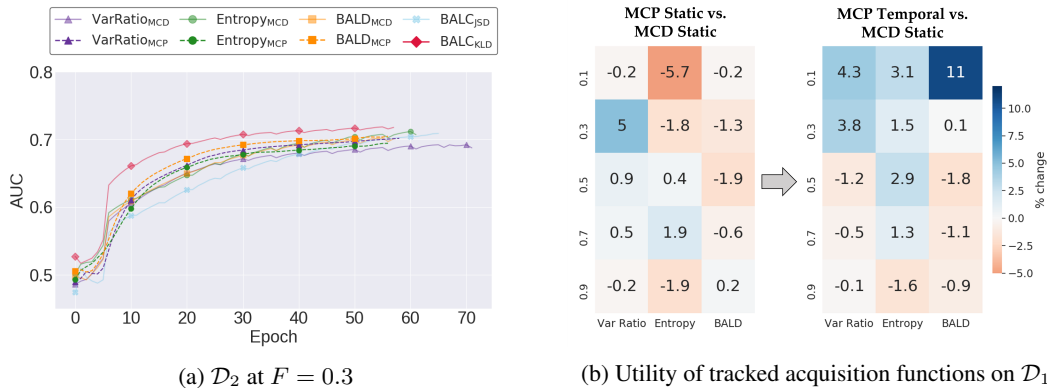
(a) $\mathcal{D}_2$ at $F = 0.3$ (b) Utility of tracked acquisition functions on $\mathcal{D}_1$

Figure 6: **Performance of active learning frameworks without an oracle. (a)** Validation AUC on $\mathcal{D}_2$ at $F = 0.3$. **(b)** Percent change in AUC when comparing MCP with static and temporal acquisition functions to MCD with static functions on $\mathcal{D}_1$. Results are averaged across five seeds. In (a), we show that BALC_{KLD} can outperform state-of-the-art acquisition functions. In (b), we show the marginal benefit of temporal acquisition functions.

noise-free oracle and explore the effect of selective oracle questioning methods. In Table 2, we present the results of these experiments across all datasets at $F = 0.1$.

We find that SoQal consistently outperforms S-response and ϵ-greedy across $\mathcal{D}_1$ - $\mathcal{D}_3$. For example, when using BALD_{MCD} on $\mathcal{D}_2$, SoQal achieves AUC = 70.7 whereas S-response and ϵ-greedy achieve AUC = 58.4 and 60.9, respectively. Such a finding suggests that SoQal is well equipped to know *when* a label should be requested from an oracle. This improved performance also coincides with

Dataset	Ac. Function α	Oracle Questioning Method		
		S-response	ϵ-greedy	SoQal
$\mathcal{D}_1$	BALD_{MCD}	49.6 (3.9)	49.1 (2.8)	**62.1** (2.1)
	BALD_{MCP}	51.7 (4.3)	50.1 (4.3)	**64.5** (1.5)
	BALC_{KLD}	54.8 (3.4)	54.8 (4.2)	**59.8** (5.5)
	Temp. BALC_{KLD}	53.6 (4.0)	52.1 (5.9)	**64.6** (6.7)
$\mathcal{D}_2$	BALD_{MCD}	58.4 (4.1)	60.9 (7.1)	**70.7** (3.8)
	BALD_{MCP}	63.8 (4.3)	63.7 (4.4)	**67.7** (4.2)
	BALC_{KLD}	58.2 (1.7)	64.3 (3.3)	**67.7** (2.4)
	Temp. BALC_{KLD}	61.2 (5.0)	60.5 (1.9)	**64.8** (5.7)
$\mathcal{D}_3$	BALD_{MCD}	58.8 (1.3)	67.3 (1.5)	**72.1** (2.5)
	BALD_{MCP}	67.6 (5.8)	66.5 (2.8)	**72.0** (4.4)
	BALC_{KLD}	62.9 (0.4)	64.3 (4.1)	**73.1** (3.3)
	Temp. BALC_{KLD}	63.0 (1.4)	65.4 (1.9)	**73.0** (2.4)
$\mathcal{D}_4$	BALD_{MCD}	48.9 (3.0)	47.4 (3.7)	46.8 (2.1)
	BALD_{MCP}	50.4 (2.6)	49.2 (2.4)	49.9 (2.9)
	BALC_{KLD}	50.4 (3.9)	47.3 (1.0)	49.5 (1.2)
	Temp. BALC_{KLD}	49.6 (2.3)	49.6 (2.3)	50.3 (1.0)

Table 2: **Test AUC of oracle questioning methods with a noise-free oracle at $F = 0.1$.** Mean (standard deviation) is presented across five random seeds for $\mathcal{D}_1 - \mathcal{D}_4$. Bold indicates top-performing method. We show that SoQal outperforms S-response and ϵ-greedy across $\mathcal{D}_1 - \mathcal{D}_3$.

reduced dependence on an oracle (see Appendix I). We also hypothesize that the poor performance of all methods on $\mathcal{D}_4$ is due to the cold-start problem (Konyushkova et al., 2017) where network learning is hindered by the limited availability of initial labelled training data. We support this claim with further experiments in Appendix I.

6.3. Active Learning with Noisy Oracle

Building on the findings in the previous section, we now explore the performance of our oracle questioning methods with a *noisy* oracle. In Fig. 7, we illustrate the test AUC on $\mathcal{D}_1$ as a function of various types and levels of noise. We also present the performance with a *noise-free* oracle (horizontal dashed lines).

We find that SoQal is more robust to a noisy oracle than ϵ-greedy and S-response. This is evident by the $\uparrow$ AUC of the former relative to the latter across different noise types and magnitudes (except at $\gamma = 0.4$ random noise). For example, at 5% random noise, SoQal achieves AUC ≈ 0.66 whereas ϵ-greedy and S-response achieve AUC ≈ 0.56 and 0.53, respectively. We also find that SoQal, in the presence of noise, continues to outperform the baseline methods in the absence of noise. For example, at 80% nearest neighbour noise, SoQal achieves AUC ≈ 0.59 whereas ϵ-greedy and S-response *without* label noise achieve AUC ≈ 0.50 and 0.52, respectively. We arrive at similar conclusions for other datasets and acquisition functions (see Appendix J). One hypothesis for this improved performance is that SoQal, by appropriately deciding when to not request a label from a noisy oracle, avoids an incorrect instance annotation, and thus allows the network to learn well.
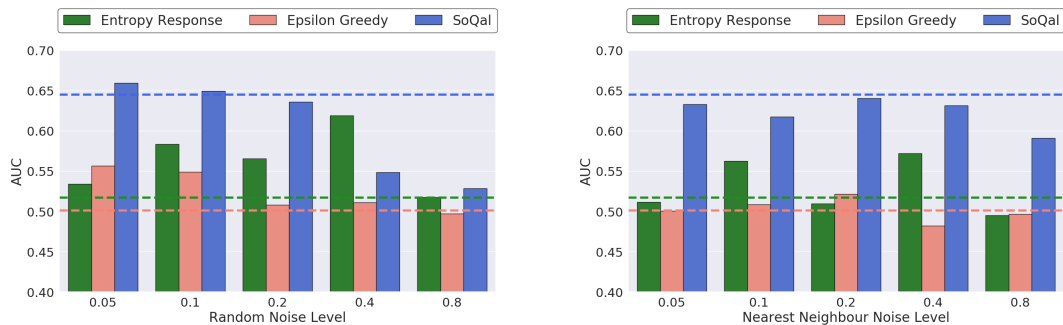
Figure 7: **Test AUC of the oracle questioning methods with (left) random and (right) nearest neighbour label noise on $\mathcal{D}_1$ while using** $\mathrm{BALD}_{\mathrm{MCP}}$. Horizontal dashed lines indicate performance of network *without* label noise. SoQal, even with a noisy oracle, outperforms baseline methods with a *noise-free* oracle.

7. Discussion

In this paper, we proposed a family of active learning frameworks which either perturb instances (Monte Carlo Perturbations) or both instances and network parameters (Bayesian Active Learning by Consistency) and observe changes in the network output distribution. We showed that BALC can outperform state-of-the-art methods such as BALD. We also found that MCP, when used alongside tracked acquisition functions, can be more favourable than MCD with static acquisition functions, particularly in low-data regimes. We also proposed SoQal, a framework that dynamically determines whether, for each acquired unlabelled instance, to request a label from an oracle or to pseudo-label it instead. We demonstrated that SoQal outperforms several baseline methods, even with a noisy oracle, while reducing a network's dependence on the oracle.

We now outline some of the limitations of our framework. In doing so, we hope to provide guidance on when it should, and should not, be used by machine learning practitioners. First, SoQal assumed that the zero-one loss incurred by the prediction network (see Fig. 4) is a reliable signal for the oracle selection network to learn from. However, the presence of class label noise hinders the reliability of this signal, introducing an error into the oracle selection process which could manifest as misplaced under- or over-dependence on an oracle. We leave it to future work to investigate the interplay between class label noise and oracle selection.

SoQal also assumed that the annotations provided by the oracle, when requested, are *consistently* reliable. However, an oracle (e.g., a physician) is likely to experience fatigue over time and exhibit undesired variability in annotation quality. Such oracle dynamics are not accounted for by our framework yet pose exciting opportunities for the future. Moreover, our framework assumed that, at most, a *single* oracle was available throughout the learning process. How-

ever, clinical settings are often characterized by the presence of multiple oracles (e.g., radiologists, cardiologists, oncologists) with different areas and levels of expertise. We hope the community considers incorporating these elements into an active learning framework which would prove quite valuable given the realistic nature of such a scenario.

Acknowledgements

References

Chow, C. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.

Clifford, G. D., Silva, I., Moody, B., Li, Q., Kella, D., Shahin, A., Kooistra, T., Perry, D., and Mark, R. G. The physionet/computing in cardiology challenge 2015: reducing false arrhythmia alarms in the icu. In *2015 Computing in Cardiology Conference*, pp. 273–276, 2015.

Clifford, G. D., Liu, C., Moody, B., Li-wei, H. L., Silva, I., Li, Q., Johnson, A., and Mark, R. G. Af classification from a short single lead ECG recording: the physionet/computing in cardiology challenge 2017. In *2017 Computing in Cardiology*, pp. 1–4, 2017.

Cohn, D., Atlas, L., and Ladner, R. Improving generalization with active learning. *Machine Learning*, 15(2): 201–221, 1994.

Cortes, C., DeSalvo, G., and Mohri, M. Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pp. 67–82. Springer, 2016.

Dekel, O., Gentile, C., and Sridharan, K. Selective sampling and active learning from single and multiple teachers. *Journal of Machine Learning Research*, 13(Sep):2655–2697, 2012.

El-Yaniv, R. and Wiener, Y. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(May):1605–1641, 2010.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international Conference on Machine Learning*, pp. 1050–1059, 2016.

Gal, Y., Islam, R., and Ghahramani, Z. Deep Bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1183–1192. JMLR. org, 2017.

Gao, M., Zhang, Z., Yu, G., Arık, S. Ö., Davis, L. S., and Pfister, T. Consistency-based semi-supervised active learning: Towards minimizing labeling cost. In *European Conference on Computer Vision*, pp. 510–526. Springer, 2020.

Geifman, Y. and El-Yaniv, R. Selectivenet: A deep neural network with an integrated reject option. In *International Conference on Machine Learning*, pp. 2151–2159. PMLR, 2019.

Gong, W., Tschiatschek, S., Turner, R., Nowozin, S., and Hernández-Lobato, J. M. Icebreaker: element-wise active information acquisition with bayesian deep latent gaussian model. *arXiv preprint arXiv:1908.04537*, 2019.

Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., and Ng, A. Y. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, 25(1):65, 2019.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Kiyasseh, D., Zhu, T., and Clifton, D. A clinical deep learning framework for continually learning from cardiac signals across diseases, time, modalities, and institutions. *Nature Communications*, 12(1):1–11, 2021a.

Kiyasseh, D., Zhu, T., and Clifton, D. Crocs: Clustering and retrieval of cardiac signals based on patient disease class, sex, and age. *Advances in Neural Information Processing Systems*, 34, 2021b.

Konyushkova, K., Sznitman, R., and Fua, P. Learning active learning from data. In *Advances in Neural Information Processing Systems*, pp. 4225–4235, 2017.

Liu, Z., Wang, Z., Liang, P. P., Salakhutdinov, R. R., Morency, L.-P., and Ueda, M. Deep gamblers: Learning to abstain with portfolio theory. In *Advances in Neural Information Processing Systems*, pp. 10622–10632, 2019.

McCallumzy, A. K. and Nigamy, K. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning*, pp. 359–367, 1998.

Pasolli, E. and Melgani, F. Active learning methods for electrocardiographic signal classification. *IEEE Transactions on Information Technology in Biomedicine*, 14(6): 1405–1416, 2010.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Settles, B. Active learning literature survey. Technical report, University of Wisconsin-Madison, Department of Computer Sciences, 2009.

Sinha, S., Ebrahimi, S., and Darrell, T. Variational adversarial active learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5972–5981, 2019.

Smailagic, A., Costa, P., Noh, H. Y., Walawalkar, D., Khandelwal, K., Galdran, A., Mirshekari, M., Fagert, J., Xu, S., Zhang, P., et al. Medal: Accurate and robust deep active learning for medical image analysis. In *IEEE International Conference on Machine Learning and Applications*, pp. 481–488, 2018.

Smailagic, A., Costa, P., Gaudio, A., Khandelwal, K., Mirshekari, M., Fagert, J., Walawalkar, D., Xu, S., Galdran, A., Zhang, P., et al. O-medal: Online active deep learning for medical image analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(4): e1353, 2020.

Urner, R., David, S. B., and Shamir, O. Learning from weak teachers. In *Artificial Intelligence and Statistics*, pp. 1252–1260, 2012.

Verma, V., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. Interpolation consistency training for semi-supervised learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3635–3641. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/504. URL https://doi.org/10.24963/ijcai.2019/504.

Wang, G., Zhang, C., Liu, Y., Yang, H., Fu, D., Wang, H., and Zhang, P. A global and updatable ecg beat classification system based on recurrent neural networks and active learning. *Information Sciences*, 501:523–542, 2019.

Watkins, C. J. C. H. Learning from delayed rewards. 1989.

Wiener, Y. and El-Yaniv, R. Agnostic selective classification. In *Advances in Neural Information Processing Systems*, pp. 1665–1673, 2011.

Wiens, J. and Guttag, J. Active learning applied to patient-adaptive heartbeat classification. *Advances in Neural Information Processing Systems*, 23, 2010.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation for consistency training. 2019.

Yan, S., Chaudhuri, K., and Javidi, T. Active learning from imperfect labelers. In *Advances in Neural Information Processing Systems*, pp. 2128–2136, 2016.

Zhang, C. and Chaudhuri, K. Active learning from weak and strong labelers. In *Advances in Neural Information Processing Systems*, pp. 703–711, 2015.

Zhu, X. J. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, Department of Computer Sciences, 2005.

A. Acquisition Functions α

$$\text{Variance Ratio} = 1 - \frac{1}{T}\sum_{t=1}^{T}\left(\delta\left(\underset{c}{\operatorname{argmax}}\, p(y = c|x, \omega_t) = \hat{c}\right)\right)$$

$$\hat{c} = \underset{c}{\operatorname{argmax}}\left(\underset{c}{\operatorname{argmax}}\, p(y = c|x, \omega_t)\; \forall t \in (1, T)\right) \tag{6}$$

where $\hat{c}$ is the most common class prediction across the T MC samples and δ is the Dirac delta function that evaluates to 1 if its argument is true, and 0 otherwise.

$$\text{Entropy H} = -\sum_{c=1}^{C} p(y = c|x) \log p(y = c|x) \tag{7}$$

$$\text{BALD} = \text{JSD}(p_1, p_2, \ldots, p_T)$$
$$= \text{H}(p(y|x)) - \mathbb{E}_{p(w|D_{train})}\left[\text{H}(p(y|x, w))\right] \tag{8}$$

where C is the number of classes in the task formulation and $p(y = c|x, \omega)$ is the probability assigned by a network parameterised by ω to a particular class c when given an input x.

B. Derivation of Monte Carlo Perturbations

$$\text{BALD}_{\text{MCP}} = \text{JSD}(p_1, p_2, \ldots, p_T)$$
$$= \text{H}(p(y|x)) - \mathbb{E}_{p(x'|D_{train})}\left[\text{H}(p(y|x, x'))\right] \tag{9}$$

$$\text{H}(p(y|x)) = \text{H}\left(\int p(y|x')p(x'|x)dx'\right)$$
$$= \text{H}\left(\int p(y|x')q_\phi(x'|x)dx'\right)$$
$$\approx \text{H}\left(\frac{1}{T}\sum_{t=1}^{T} p(y|\hat{x}'_t)\right) \tag{10}$$

where x' represents the perturbed input, T is the number of Monte Carlo samples, and $\hat{x}'_t \sim q_\phi(x'|x)$ is a sample from some perturbation generator.

$$\mathbb{E}_{p(x'|D_{train})}\left[\text{H}(p(y|x, x'))\right] = \mathbb{E}_{q_\phi(x'|x)}\left[\text{H}(p(y|x, x'))\right]$$
$$\approx \frac{1}{T}\sum_{t=1}^{T}\left[\text{H}(p(y|\hat{x}'_t))\right]$$
$$= \frac{1}{T}\sum_{t=1}^{T}\left[-\sum_{c=1}^{C} p(y = c|\hat{x}'_t) \log p(y = c|\hat{x}'_t)\right] \tag{11}$$

C. Derivation of Bayesian Active Learning by Consistency

$$\text{BALC}_{\text{JSD}} = \mathbb{E}_{p(\omega|D_{train})} \left[\mathcal{D}_{KL}(p(y|x,\omega) \| p(y|x',\omega)) \right] - \mathcal{D}_{KL}(p(y|x) \| p(y|x')) \tag{12}$$

where x' is the perturbed version of the input and $\mathcal{D}_{KL}$ is the Kullback-Leibler divergence.

$$\begin{aligned}
\mathbb{E}_{p(\omega|D_{train})} \left[\mathcal{D}_{KL}(p(y|x,\omega) \| p(y|x',\omega)) \right] &= \mathbb{E}_{q_\theta(\omega)} \left[\mathcal{D}_{KL}(p(y|x,\omega) \| p(y|x',\omega)) \right] \\
&\approx \frac{1}{T} \sum_{t=1}^{T} \left[\mathcal{D}_{KL}(p(y|x,\hat{\omega}_t \| p(y|\hat{x},\hat{\omega}_t)) \right] \\
&= \frac{1}{T} \sum_{t=1}^{T} \left[\sum_{c=1}^{C} p(y=c|x,\hat{\omega}_t) \log \frac{p(y=c|x,\hat{\omega}_t)}{p(y=c|x',\hat{\omega}_t)} \right]
\end{aligned} \tag{13}$$

$$\begin{aligned}
\mathcal{D}_{KL}(p(y|x)\|p(y|x')) &= \mathcal{D}_{KL}\left(\int p(y|\omega,x)p(\omega)d\omega \,\Big\|\, \int p(y|\omega,x')p(\omega)d\omega \right) \\
&= \mathcal{D}_{KL}\left(\int p(y|\omega,x)q_\theta(\omega)d\omega \,\Big\|\, \int p(y|\omega,x')q_\theta(\omega)d\omega \right) \\
&\approx \mathcal{D}_{KL}\left(\frac{1}{T} \sum_{t=1}^{T} p(y|\hat{\omega}_t,x) \,\Big\|\, \frac{1}{T} \sum_{t=1}^{T} p(y|\hat{\omega}_t,x') \right) \\
&= \frac{1}{C} \sum_{c=1}^{C} \left[\frac{1}{T} \sum_{t=1}^{T} p(y=c|\hat{\omega}_t,x) \log \frac{\frac{1}{T}\sum_{t=1}^{T} p(y=c|\hat{\omega}_t,x)}{\frac{1}{T}\sum_{t=1}^{T} p(y=c|\hat{\omega}_t,\hat{x})} \right]
\end{aligned} \tag{14}$$

where the integral is approximated by T Monte Carlo samples, $\hat{\omega} \sim q_\theta(w)$ represents the parameters sampled from the Monte Carlo distribution, and C represents the number of classes in the task formulation.

D. Datasets

D.1. Data Preprocessing

Each dataset consists of cardiac time-series waveforms alongside their corresponding cardiac arrhythmia label. Each waveform was split into non-overlapping frames of 2500 samples.

PhysioNet 2015 PPG, $\mathcal{D}_1$ (Clifford et al., 2015). This dataset consists of photoplethysmogram (PPG) time-series waveforms sampled at 250Hz and five cardiac arrhythmia labels: Asystole, Extreme Bradycardia, Extreme Tachycardia, Ventricular Tachycardia, and Ventricular Fibrillation. Only patients with a True Positive Alarm are considered. The PPG frames were normalized in amplitude between the values of 0 and 1.

PhysioNet 2015 ECG, $\mathcal{D}_2$ (Clifford et al., 2015). This dataset consists of electrocardiogram (ECG) time-series waveforms sampled at 250Hz and five cardiac arrhythmia labels: Asystole, Extreme Bradycardia, Extreme Tachycardia, Ventricular Tachycardia, and Ventricular Fibrillation. Only patients with a True Positive Alarm are considered. The ECG frames were normalized in amplitude between the values of 0 and 1.

PhysioNet 2017 ECG, $\mathcal{D}_3$ (Clifford et al., 2017). This dataset consists of ECG time-series waveforms sampled at 300Hz and four labels: Normal, Atrial Fibrillation, Other, and Noisy. The ECG frames were not normalized.

Cardiology ECG, $\mathcal{D}_4$ (Hannun et al., 2019). This dataset consists of ECG time-series waveforms sampled at 200Hz and twelve cardiac arrhythmia labels: Atrial Fibrillation, Atrio-ventricular Block, Bigeminy, Ectopic Atrial Rhythm, Idioventricular Rhythm, Junctional Rhythm, Noise, Sinus Rhythm, Supraventricular Tachycardia, Trigeminy, Ventricular Tachycardia, and Wenckebach. Sudden bradycardia cases were excluded from the data as they were not included in the original formulation by the authors. The ECG frames were not normalized.

D.2. Data Samples

All datasets were split into training, validation, and test sets according to patient ID using a 60, 20, 20 configuration. In other words, patients appeared in only one of the sets. Samples in the training set were further split into a labelled and an unlabelled subset, also according to patient ID. In Tables 3 and 4, we show the number of samples and patients used in each of these sets.

D.2.1. CONSISTENCY-BASED ACTIVE LEARNING EXPERIMENTS

Table 3: Sample sizes (number of patients for cardiac datasets) of train/val/test splits. Datasets $\mathcal{D}_1$ to $\mathcal{D}_4$ are defined in the main manuscript.

Dataset	Fraction F	Train Labelled	Train Unlabelled	Val	Test
$\mathcal{D}_1$	0.1	401 (18)	4,233 (171)	1,124 (47)	1,435 (58)
	0.3	1,285 (55)	3,349 (134)		
	0.5	2,187 (92)	2,447 (97)		
	0.7	3,132 (129)	1,502 (60)		
	0.9	4,184 (166)	450 (23)		
$\mathcal{D}_2$	0.1	401 (18)	4,233 (171)	1,124 (47)	1,435 (58)
	0.3	1,285 (55)	3,349 (134)		
	0.5	2,187 (92)	2,447 (97)		
	0.7	3,132 (129)	1,502 (60)		
	0.9	4,184 (166)	450 (23)		
$\mathcal{D}_3$	0.1	1,776 (545)	16,479 (4,914)	4,582 (1,364)	5,824 (1705)
	0.3	5,399 (1636)	12,856 (3,823)		
	0.5	9,054 (2727)	9,201 (2,732)		
	0.7	12,733 (3818)	5,522 (1,641)		
	0.9	16,365 (4909)	1,890 (550)		
$\mathcal{D}_4$	0.1	452 (20)	4,110 (181)	1,131 (50)	1,386 (62)
	0.3	1,368 (60)	3,194 (141)		
	0.5	2,280 (101)	2,282 (100)		
	0.7	3,200 (140)	1,362 (61)		
	0.9	4,079 (180)	483 (21)		

D.2.2. SELECTIVE ORACLE QUESTIONING EXPERIMENTS

Table 4: Sample sizes (number of patients) of training, validation, and test sets.

Dataset	Training Labelled	Training Unlabelled	Validation	Test
$\mathcal{D}_1$	401 (18)	4233 (171)	1124 (47)	1435 (58)
$\mathcal{D}_2$	401 (18)	4233 (171)	1124 (47)	1435 (58)
$\mathcal{D}_3$	1,776 (545)	16,479 (4,914)	4,582 (1,364)	5,824 (1,705)
$\mathcal{D}_4$	452 (20)	4,110 (181)	1,131 (50)	1,386 (62)

E. Implementation Details

In this section, we outline the network architecture used for all experiments conducted in the main manuscript. We also outline the batchsize and learning rate associated with training on each of the datasets.

E.1. Network Architecture

Table 5: Network architecture used for time-series experiments. K, C_{in}, and C_{out} represent the kernel size, number of input channels, and number of output channels, respectively. A stride of 3 was used for Conv1D operators, respectively.

(a) Network for time-series datasets

Layer Number	Layer Components	Kernel Dimension
1	Conv 1D BatchNorm ReLU MaxPool(2) Dropout(0.1)	7 x 1 x 4 (K x C_{in} x C_{out})
2	Conv 1D BatchNorm ReLU MaxPool(2) Dropout(0.1)	7 x 4 x 16
3	Conv 1D BatchNorm ReLU MaxPool(2) Dropout(0.1)	7 x 16 x 32
4	Linear ReLU	320 x 100
5	Linear	100 x C (classes)

E.2. Experiment Details

Table 6: Batchsize and learning rates used for training with different datasets. The Adam optimizer was used for all experiments.

Dataset	Batchsize	Learning Rate
$\mathcal{D}_1$	256	10^{-4}
$\mathcal{D}_2$	256	10^{-4}
$\mathcal{D}_3$	256	10^{-4}
$\mathcal{D}_4$	16	10^{-4}

E.3. Perturbation Details

When conducting the MCP and BALC experiments, we perturbed each of the time-series frames with additive Gaussian noise, $\epsilon \sim \mathcal{N}(0, \sigma)$ where we chose σ based on the specific dataset to avoid introducing too much noise. The details of these perturbations can be found in Table 7. We applied all perturbations to the input data before normalization.

Table 7: Perturbations applied to different datasets during MCP and BALC implementations. p represents the probability of applying a particular augmentation method.

Dataset	Perturbation
$\mathcal{D}_1$	$\epsilon \sim \mathcal{N}(0, 100)$
$\mathcal{D}_2$	$\epsilon \sim \mathcal{N}(0, 100)$
$\mathcal{D}_3$	$\epsilon \sim \mathcal{N}(0, 100)$
$\mathcal{D}_4$	$\epsilon \sim \mathcal{N}(0, 100)$

E.4. Baseline Implementations

In this section, we outline our implementation of the baseline methods used in the selective oracle questioning experiments.

E.4.1. ENTROPY RESPONSE, (S-RESPONSE)

This approach is anchored around the idea that network outputs that exhibit high entropy (i.e., close to a uniform distribution) are likely to correspond to instances that the network is uncertain of. Consequently, we exploited this idea to determine whether a label is requested from an oracle or if a pseudo-label should be generated instead. More specifically, we introduced a threshold, $S_{Entropy} = w \times S_{Max}$, which is a fraction of the maximum entropy possible for a particular classification problem. As mentioned, $S_{Max} = \log C$, where C is the number of classes. We chose $w = 0.9$ to balance between oracle dependence and pseudo-label accuracy. This value was kept fixed during training. In our implementation, we take the mean of the network outputs as a result of the perturbations, calculate its entropy, and determine whether it exceeds the aforementioned threshold. If it does, then the uncertainty is deemed high and a label is requested from an oracle.

E.4.2. EPSILON GREEDY, (ϵ-GREEDY)

This approach is inspired by the reinforcement learning literature and is used to decay the dependence of network on the oracle. More specifically, we define $\epsilon = e^{\frac{-epoch}{k \times \tau}}$ where epoch represents the training epoch number and τ is the epoch interval at which acquisitions are performed. ϵ decays from $1 \to 0$ as training progresses. We chose $k = \tau = 5$ in order to balance between oracle dependence and pseudo-label accuracy. To determine whether a label is requested from an oracle, we generate a random number, $R \sim \mathcal{U}(0, 1)$, for a uniform distribution and check whether it is below ϵ. If this is satisfied, then an oracle is requested for a label, and a pseudo-label is generated otherwise. As designed, this approach starts off with 100% dependence on an oracle and decays towards minimal dependence as training progresses.

F. Pseudo-code

We elucidate the entire active learning framework in Algorithms 1 and 2 where the coloured lines indicate the components associated with the (optional) tracking of acquisition functions. It is worthwhile to note that our oracle selection framework is independent of the acquisition of unlabelled instances. As such, it is flexible enough to be used alongside other acquisition functions.

Algorithm 1 Bayesian Active Learning by Consistency

Input: acquisition epochs τ, temporal period Δt, labelled data $\mathcal{D}_L$, unlabelled data $\boldsymbol{X}_U$, network parameters $\boldsymbol{\theta}\ \boldsymbol{\omega}\ \boldsymbol{\phi}$, MC samples T, acquisition fraction b

1: **while** training **do**
2:     **if** epoch in Δt **then**
3:         **for** $\boldsymbol{x} \sim X_U$ **do**
4:             $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\epsilon}, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$
5:             **for** t-th MC sample in T **do**
6:                 obtain $p(y|\boldsymbol{x}, \boldsymbol{\theta}_t)$ ▷ original input
7:                 obtain $p(y|\boldsymbol{x}', \boldsymbol{\theta}_t)$ ▷ perturbed input
8:             calculate $\alpha(\boldsymbol{x})$ using (2) or (1)
9:             $\alpha(\boldsymbol{x}, t) = \alpha$
10:     **if** epoch in τ **then**   ▷ acquire unlabelled instances
11:         calculate α using (3)
12:         SortDescending(α)
13:         $\boldsymbol{x_b} \subset \boldsymbol{X_U}$
14:         $y_b = \text{SoQal}(\boldsymbol{x_b})$ ▷ selective oracle questioning
15:         $\boldsymbol{X_U} \in (\boldsymbol{X_U} \setminus (\boldsymbol{x_b}, y_b))$
16:         $\mathcal{D}_L \in (\mathcal{D}_L \cup (\boldsymbol{x_b}, y_b))$

Algorithm 2 SoQal

Input: unlabelled instances $\boldsymbol{x_b}$, Hellinger distance D_H, Hellinger threshold S

1: **for** $\boldsymbol{x}_u \sim \boldsymbol{x_b}$ **do**
2:     $r_u = g_\phi(\boldsymbol{x}_u)$
3:     **if** $D_H > S$ **then**
4:         calculate $p(A)$ from (5)
5:         **if** $p(A) = 1$ **then**
6:             $y_b \subset \boldsymbol{Y_U}$ ▷ request label from physician
7:         **else**
8:             $y_b = \underset{c}{\arg\max}\ p_\omega(y = c|\boldsymbol{x}_u)$ ▷ pseudo-label

---

G. Validation Set AUC with Non-Temporal Acquisition Functions in the Absence of Oracle

In the main manuscript, we presented a subset of results for experiments in which oracles are absent and thus unavailable to provide annotations. Instead, unlabelled instances are pseudo-labelled based on network-generated predictions. In this section, we include an exhaustive set of results for all those experiments. More specifically, we illustrate in Figs. 8 - 11 the validation AUC of the various AL methods for datasets $\mathcal{D}_1$ - $\mathcal{D}_5$. At a high level and across datasets, we find that the cold-start problem is likely to occur at low fraction values ($\beta = 0.1$). We include more details in the respective sections.

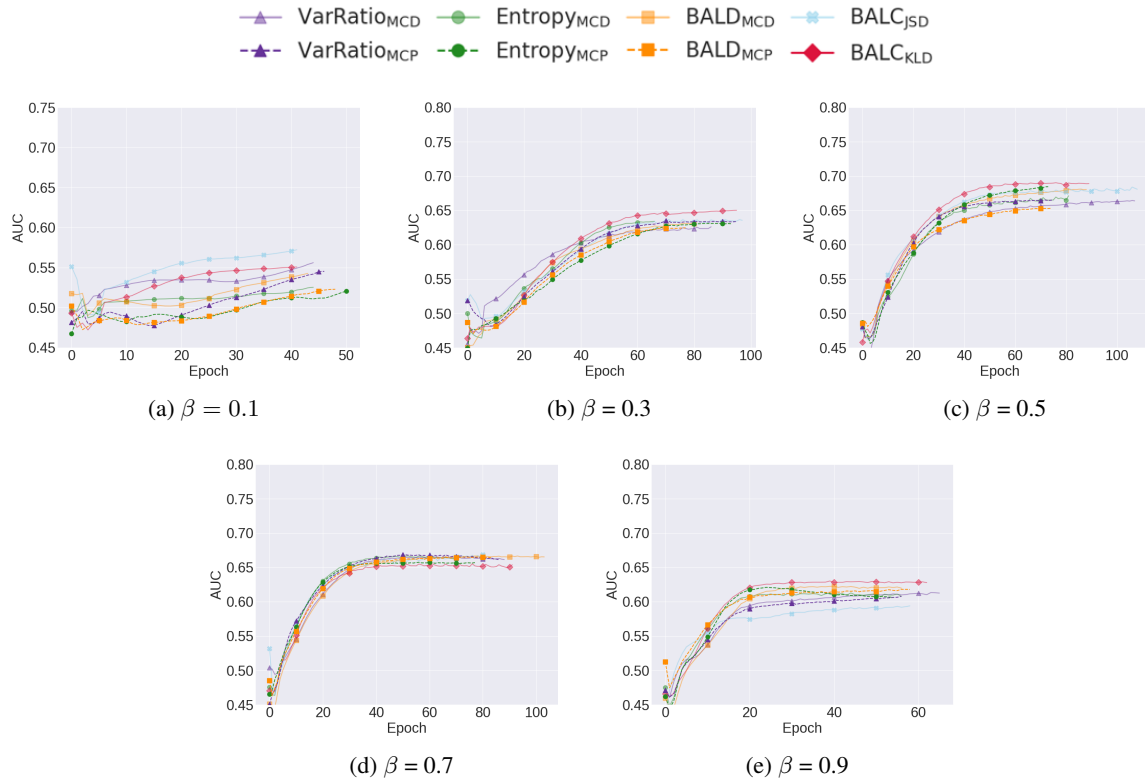## G.1. PhysioNet 2015 PPG, $\mathcal{D}_1$

Figure 8: Mean validation set AUC for the various methodologies and acquisition functions on $\mathcal{D}_1$ at increasing fraction levels $\beta = (0.1, 0.3, 0.5, 0.7, 0.9)$. The no-oracle cold-start problem is observed at $\beta = 0.1$ where active learning approaches fail due to few available labelled training instances. Clear benefits of our methods can be seen at $\beta = 0.5, 0.7$. Results are averaged across 5 seeds.
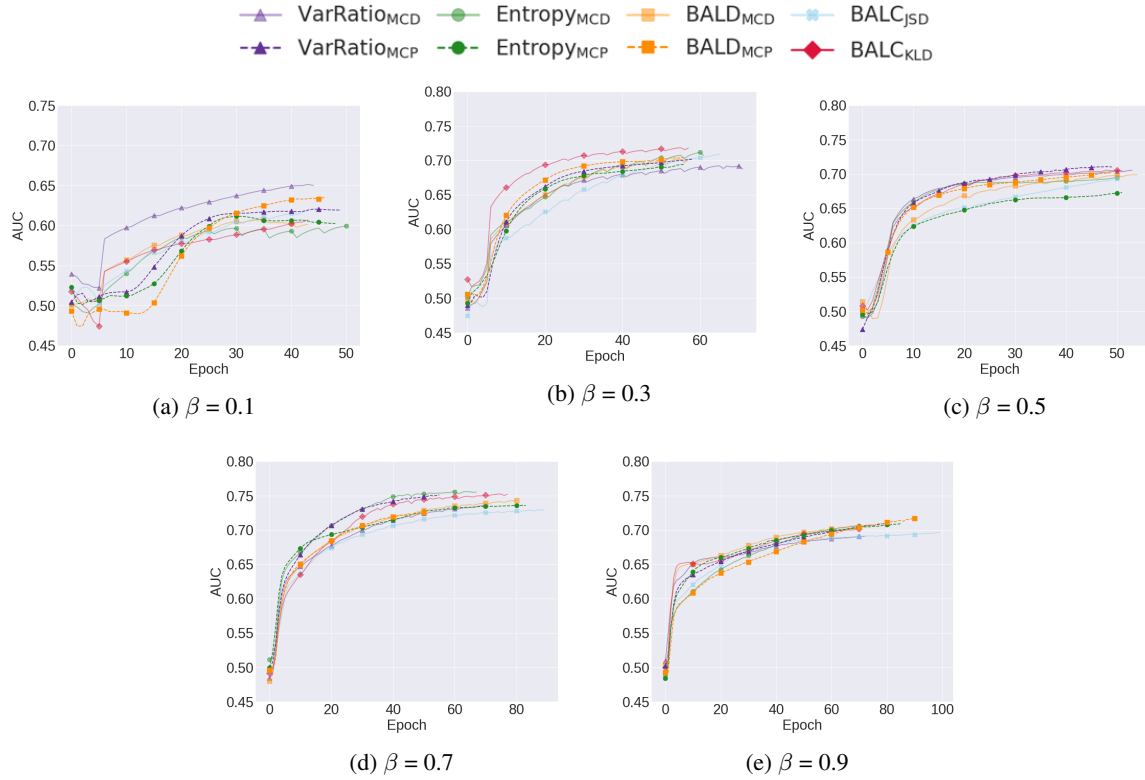
G.2. PhysioNet 2015 ECG, $\mathcal{D}_2$

(a) $\beta = 0.1$

(b) $\beta = 0.3$

(c) $\beta = 0.5$

(d) $\beta = 0.7$

(e) $\beta = 0.9$

Figure 9: Mean validation set AUC for the various methodologies and acquisition functions on $\mathcal{D}_2$ at increasing fraction levels $\beta = (0.1, 0.3, 0.5, 0.7, 0.9)$. Our methods include MCP and BALC methods. The no-oracle cold-start problem is observed at $\beta = 0.1$ where active learning approaches fail due to few available labelled training instances. However, our approaches outperform all others at $\beta = 0.3, 0.5, 0.7, 0.9$. Results are averaged across 5 seeds.
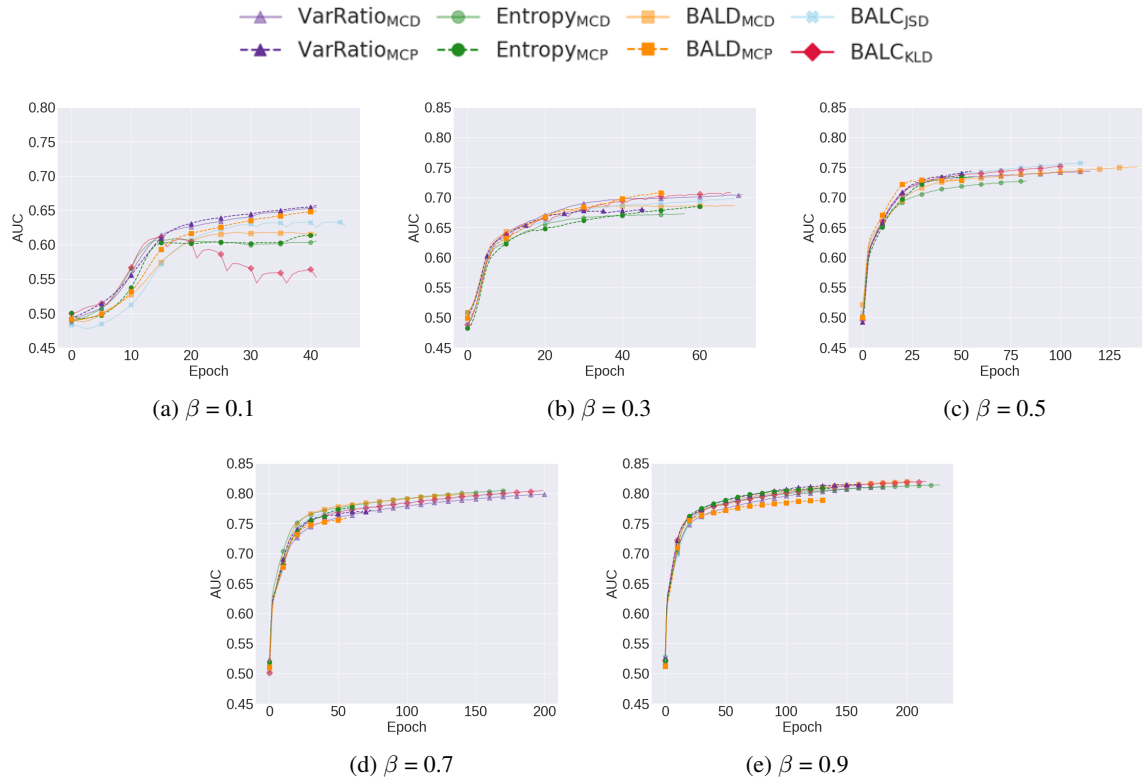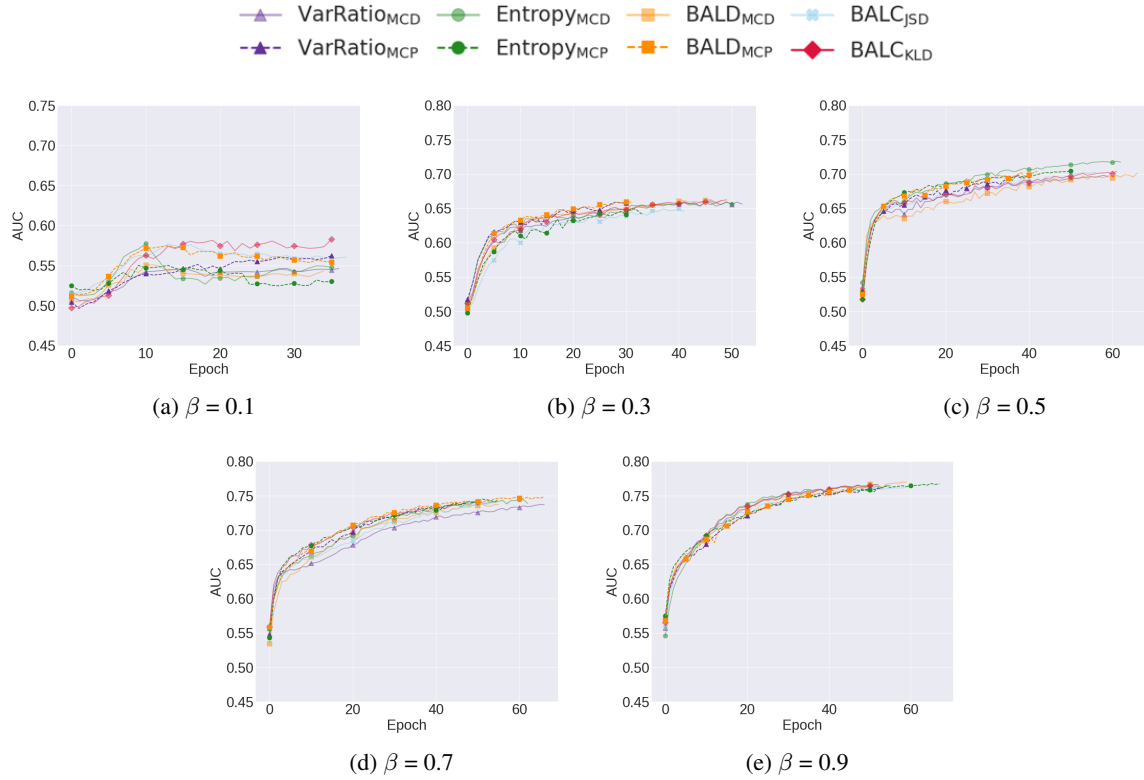
G.3. PhysioNet 2017 ECG, $\mathcal{D}_3$

Figure 10: Mean validation set AUC for the various methodologies and acquisition functions on $\mathcal{D}_3$ at increasing fraction levels $\beta = (0.1, 0.3, 0.5, 0.7, 0.9)$. Our methods include MCP and BALC methods. The no-oracle cold-start problem is observed at $\beta = 0.1$ where active learning approaches fail due to few available labelled training instances. Most methods perform on par with the no active learning strategy for this particular dataset. Results are averaged across 5 seeds.

G.4. Cardiology ECG, $\mathcal{D}_4$

Figure 11: Mean validation set AUC for the various methodologies and acquisition functions on $\mathcal{D}_4$ at increasing fraction levels $\beta = (0.1, 0.3, 0.5, 0.7, 0.9)$. Our methods include MCP and BALC methods. The no-oracle cold-start problem is *not* observed for this dataset. Most methods perform comparably to one another at high values of β. Results are averaged across 5 seeds.

H. Effect of MCP with Tracked Acquisition Functions on Performance

In this section, we are interested in quantifying the effect of implementing a temporal acquisition function in conjunction with MCP on performance. In Fig. 12, we illustrate two columns of matrices. The first column reflects the percent change in generalization performance between implementing MCP and MCD with static temporal functions (i.e., without tracking) for three different datasets.

We find that there are mixed results. For example, on $mathcalD_2$ at $\beta = 0.5$, BALD_{MCP} outperforms BALD_{MCD} by 6.5%. However, on $mathcalD_4$ at $\beta = 0.5$, $\text{Entropy}_{\text{MCP}}$ performs worse than $\text{Entropy}_{\text{MCD}}$ by 6.7%. Furthermore, upon applying tracked acquisition functions, we also obtain mixed results. In many cases, there are notable improvements. For example, on $mathcalD_3$ at $\beta = 0.5$, Temporal $\text{Entropy}_{\text{MCP}}$ improves performance by an additional $0.3 + 2.3 = 2.6\%$. On the other hand, at $\beta = 0.7$, Temporal $\text{Entropy}_{\text{MCP}}$ worsens performance by 2.1%. Based on these findings, we would recommend that the utility of temporal acquisition functions be determined on a case-by-case basis.
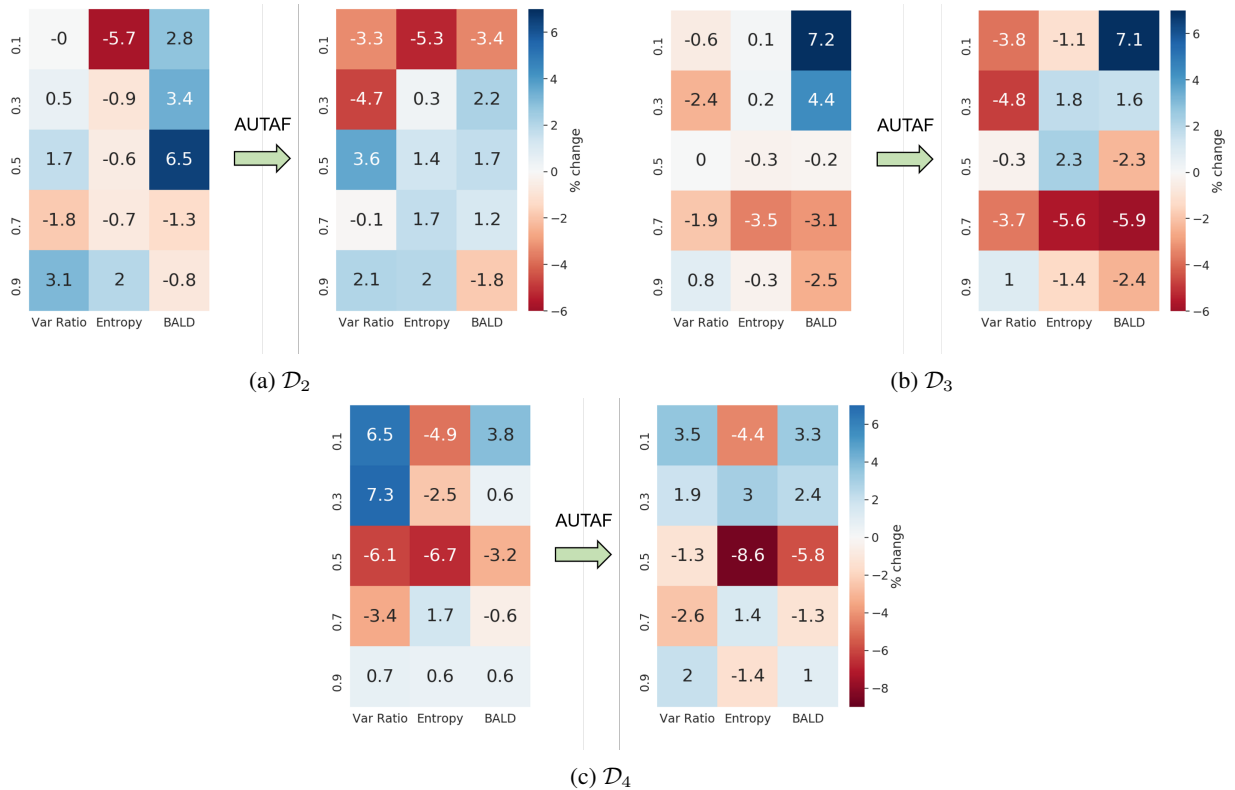

Figure 12: Mean percent change in test AUC when comparing MCP with static and tracked acquisition functions to MCD with their static counterparts on **(a)** $\mathcal{D}_2$ and **(b)** $\mathcal{D}_3$ and **(c)** $\mathcal{D}_4$. We show results for Var Ratio, Entropy, and BALD, at all fractions, $\beta \in [0.1, 0.3, 0.5, 0.7, 0.9]$.

I. Dependence of SoQal on Oracle

Recall that one of our original goals is to reduce the labelling burden that is placed on an oracle (e.g., physician). In this section, we look to quantify the degree to which our active learning framework is dependent on an oracle. To do so, we define a metric, which we refer to as the oracle ask rate (OAR), that quantifies the proportion of all acquired instances whose labels are requested from an oracle. For example, an $\text{OAR} = 50\%$ implies that 50% of the acquired instances had labels provided by an oracle, with the remaining 50% pseudo-labelled by the network. In Fig. 13a (left), we present the oracle ask rate of SoQal as a function of the type and magnitude of label noise.

I.1. Label noise and dependence on oracle

In Fig. 13a (left), we find that SoQal alleviates the labelling burden placed on an oracle. For example, SoQal when used in conjunction with BALD_{MCP} and without label noise, achieves $\text{OAR} \approx 0.68 < 1.0$. This suggests that 32% fewer label requests were sent to an oracle compared to an active learning framework that requests 100% of its labels from an oracle. However, these fewer label requests do result in slightly lower performance compared to the 100% method (see earlier Table 2). This is expected since noise-free oracle-provided labels are likely to be of higher quality than those generated by a network (via pseudo-labels).

We also find that SoQal can become more dependent on an oracle with increased levels of noise. For example, with random
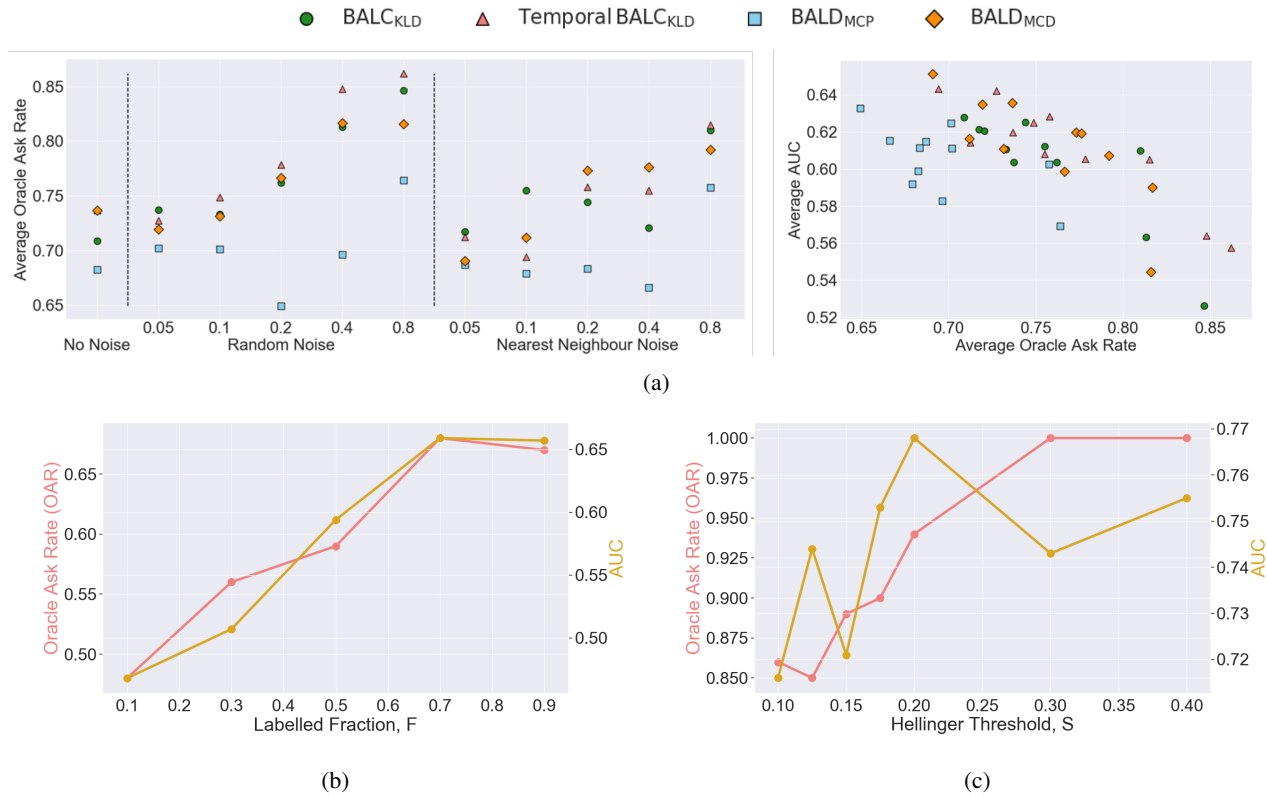

Figure 13: **Dependence of SoQal on oracle as a function label noise, data availability, and the Hellinger threshold.** **(a)** (left) Average oracle ask rate (OAR) as a function of various types and levels of label noise, shown for a subset of acquisition functions. (right) Correlation between oracle ask rate and generalization performance. We also present the OAR and corresponding AUC while using BALD_{MCD} as a function of **(b)** the labelled fraction, F, of labelled data and **(c)** the Hellinger threshold, S. All results are averaged across five random seeds. In (a) (right), we show that the reduced dependence of a network on an oracle can be advantageous. This is particularly true if the oracle exhibits label noise. In (b), we show that increasing the amount of labelled data increases the network's dependence on an oracle. In (c), we show that tuning S provides researchers with flexibility over how often to request a label from an oracle.

noise $\gamma = 0.05 \rightarrow 0.8$, the average OAR $\approx 0.72 \rightarrow 0.80$. This increased dependence can be problematic, particularly when the oracle is highly likely to annotate unlabelled instances incorrectly. Nonetheless, SoQal, even with extreme label noise, achieves OAR < 1.0 (fewer requests) while continuing to outperform the remaining questioning methods (see earlier Fig. 7).

In Fig. 13a (right), we further explore the relationship between the oracle ask rate and generalization performance. We find that, with label noise, the decreased dependence of SoQal on an oracle is associated with improved generalization performance. This is shown by the negative correlation between the average OAR and the AUC. For example, with OAR $\approx 0.85 \rightarrow 0.65$, the AUC $\approx 0.53 \rightarrow 0.63$. This suggests that SoQal performs better while requesting *fewer* labels. Such a finding reaffirms the observation that SoQal appropriately learns *when* to request a label from an oracle or to pseudo-label instead.

I.2. Data availability and dependence on oracle

The previous oracle questioning results we presented were based on experiments conducted on limited, labelled datasets (i.e., $F = 0.1$). In this section, we explore the dependence of SoQal on an oracle when provided with more labelled data (i.e., $F > 0.1$). In Fig. 13b, we present the OAR and corresponding AUC as a function of the amount of available labelled data, F. As expected, we find that performance improves with more data. For example, as $F = 0.1 \rightarrow 0.9$, AUC $\approx 0.50 \rightarrow 0.65$. We also find that SoQal continues to alleviate the labelling burden placed on an oracle even when more labelled data are available. For example, at $F = 0.9$, SoQal exhibits OAR $\approx 0.67 < 1.0$, reflecting a 33% reduction in labelling burden.

I.3. Hellinger threshold and dependence on oracle

We claimed that the Hellinger threshold, S, can be tuned according to the relative trust one has in the network and the oracle. A higher threshold ($\uparrow S$) implies that less trust is placed in the network than in the oracle. In Fig. 13c, we present the OAR and corresponding AUC as a function of the Hellinger threshold, S.

Consistent with expectations, we find that the dependence of SoQal on the oracle increases with the threshold. For example, as $S = 0.10 \rightarrow 0.40$, OAR $\approx 0.86 \rightarrow 1.0$. Such a finding suggests that researchers can set the threshold, S, a priori based on the extent to which they would like to request labels from an oracle. We also find that this increased dependence sometimes results in worse performance. For example, although transitioning from $S = 0.20 \rightarrow 0.30$ leads to OAR $\approx 0.95 \rightarrow 1.0$, the AUC $\approx 0.77 \rightarrow 0.74$. We hypothesize that this worse performance is due to inherent label noise in the datasets. Therefore, a strategy which depends on an oracle 100% of the time is worse than one which delegates some of the labelling to the network instead. Such a finding provides further evidence in support of a dynamic oracle selection strategy.

J. Performance of Oracle Selection Strategies with Noisy Oracle

Over-reliance on an oracle could be detrimental for an active learning algorithm if that oracle is unable to label instances accurately. In the case of physicians, this inability could arise due to poor training, fatigue, or the difficulty of a particular case being diagnosed. We simulate these scenarios by injecting label noise of various magnitude into the datasets. In this section, we illustrate the performance of three oracle selection strategies, SoQal, Epsilon Greedy, and Entropy Response, in response to label noise. The results are shown for random and nearest neighbour noise in Secs. J.1 and J.2, respectively.

J.1. Label Noise - Random

Although random noise can be considered an extreme case, it is nonetheless plausible in certain scenarios where labellers are poorly trained or the task at hand is too difficult. In this section, we illustrate, in Tables 8a - 8c, the degree to which the test AUC is affected by the introduction of random label noise during the active learning procedure.

As expected, extreme levels of noise negatively affect performance. For instance, this can be seen in Table 8a at $\mathcal{D}_2$ using BALD_{MCD} where increasing the level of random noise from $5\% \rightarrow 80\%$ leads to a reduction of $\text{AUC} = 0.679 \rightarrow 0.556$. Across most noise levels, SoQal continues to outperform Epsilon Greedy and Entropy Response. This finding is consistent with that presented in the main manuscript and illustrates the relative robustness of SoQal to label noise.

Table 8: Mean test AUC of oracle questioning strategies as a function of increasing levels of *random* label noise by the oracle. Results are shown for datasets $\mathcal{D}_1 - \mathcal{D}_5$ and all acquisition functions. Mean and standard deviation values are shown across five seeds.

(a) SoQal

Dataset	Ac. Function α	Random Noise Level				
		0.05	0.10	0.20	0.40	0.80
$\mathcal{D}_1$	BALD_{MCD}	0.595 ± 0.053	0.554 ± 0.028	0.558 ± 0.042	0.600 ± 0.029	0.511 ± 0.055
	BALD_{MCP}	0.659 ± 0.014	0.650 ± 0.027	0.636 ± 0.029	0.549 ± 0.058	0.528 ± 0.029
	BALC_{KLD}	0.564 ± 0.058	0.570 ± 0.045	0.562 ± 0.067	0.498 ± 0.038	0.477 ± 0.011
	Temporal BALC_{KLD}	0.634 ± 0.026	0.597 ± 0.035	0.611 ± 0.040	0.494 ± 0.034	0.490 ± 0.026
$\mathcal{D}_2$	BALD_{MCD}	0.679 ± 0.017	0.659 ± 0.042	0.646 ± 0.044	0.602 ± 0.047	0.556 ± 0.065
	BALD_{MCP}	0.643 ± 0.020	0.677 ± 0.053	0.637 ± 0.042	0.619 ± 0.033	0.602 ± 0.041
	BALC_{KLD}	0.652 ± 0.037	0.659 ± 0.056	0.649 ± 0.054	0.614 ± 0.016	0.522 ± 0.032
	Temporal BALC_{KLD}	0.655 ± 0.048	0.701 ± 0.029	0.628 ± 0.074	0.594 ± 0.041	0.581 ± 0.032
$\mathcal{D}_3$	BALD_{MCD}	0.750 ± 0.017	0.742 ± 0.031	0.718 ± 0.037	0.646 ± 0.023	0.584 ± 0.017
	BALD_{MCP}	0.724 ± 0.022	0.707 ± 0.021	0.682 ± 0.038	0.629 ± 0.029	0.537 ± 0.025
	BALC_{KLD}	0.724 ± 0.032	0.725 ± 0.028	0.702 ± 0.024	0.651 ± 0.046	0.564 ± 0.040
	Temporal BALC_{KLD}	0.725 ± 0.031	0.738 ± 0.013	0.705 ± 0.017	0.596 ± 0.071	0.546 ± 0.041
$\mathcal{D}_4$	BALD_{MCD}	0.506 ± 0.019	0.479 ± 0.022	0.496 ± 0.023	0.490 ± 0.010	0.518 ± 0.029
	BALD_{MCP}	0.499 ± 0.037	0.508 ± 0.022	0.523 ± 0.027	0.495 ± 0.023	0.503 ± 0.021
	BALC_{KLD}	0.491 ± 0.026	0.481 ± 0.023	0.496 ± 0.031	0.518 ± 0.012	0.525 ± 0.011
	Temporal BALC_{KLD}	0.522 ± 0.016	0.505 ± 0.027	0.501 ± 0.021	0.511 ± 0.025	0.515 ± 0.031

(b) Epsilon Greedy

Dataset	Ac. Function α	Random Noise Level				
		0.05	0.10	0.20	0.40	0.80
$\mathcal{D}_1$	$BALD_{MCD}$	0.496 ± 0.058	0.494 ± 0.029	0.476 ± 0.030	0.507 ± 0.044	0.501 ± 0.056
	$BALD_{MCP}$	0.557 ± 0.018	0.549 ± 0.036	0.508 ± 0.032	0.511 ± 0.033	0.497 ± 0.057
	$BALC_{KLD}$	0.517 ± 0.014	0.518 ± 0.035	0.504 ± 0.028	0.506 ± 0.034	0.498 ± 0.017
	Temporal $BALC_{KLD}$	0.525 ± 0.037	0.512 ± 0.040	0.501 ± 0.025	0.493 ± 0.019	0.497 ± 0.039
$\mathcal{D}_2$	$BALD_{MCD}$	0.600 ± 0.053	0.628 ± 0.053	0.589 ± 0.037	0.612 ± 0.022	0.555 ± 0.041
	$BALD_{MCP}$	0.629 ± 0.023	0.614 ± 0.048	0.536 ± 0.081	0.575 ± 0.029	0.588 ± 0.050
	$BALC_{KLD}$	0.619 ± 0.038	0.586 ± 0.054	0.629 ± 0.061	0.613 ± 0.045	0.582 ± 0.067
	Temporal $BALC_{KLD}$	0.630 ± 0.041	0.652 ± 0.029	0.579 ± 0.034	0.610 ± 0.036	0.564 ± 0.035
$\mathcal{D}_3$	$BALD_{MCD}$	0.663 ± 0.018	0.661 ± 0.011	0.632 ± 0.021	0.626 ± 0.012	0.588 ± 0.017
	$BALD_{MCP}$	0.671 ± 0.017	0.670 ± 0.016	0.639 ± 0.024	0.623 ± 0.019	0.574 ± 0.041
	$BALC_{KLD}$	0.665 ± 0.022	0.650 ± 0.014	0.664 ± 0.013	0.618 ± 0.034	0.595 ± 0.031
	Temporal $BALC_{KLD}$	0.661 ± 0.012	0.651 ± 0.018	0.651 ± 0.016	0.629 ± 0.019	0.612 ± 0.049
$\mathcal{D}_4$	$BALD_{MCD}$	0.473 ± 0.030	0.480 ± 0.033	0.469 ± 0.024	0.468 ± 0.018	0.493 ± 0.015
	$BALD_{MCP}$	0.508 ± 0.016	0.495 ± 0.019	0.498 ± 0.043	0.494 ± 0.032	0.497 ± 0.015
	$BALC_{KLD}$	0.492 ± 0.026	0.496 ± 0.021	0.481 ± 0.025	0.491 ± 0.020	0.498 ± 0.021
	Temporal $BALC_{KLD}$	0.514 ± 0.017	0.528 ± 0.017	0.500 ± 0.008	0.498 ± 0.033	0.504 ± 0.037

(c) Entropy Response

Dataset	Ac. Function α	Random Noise Level				
		0.05	0.10	0.20	0.40	0.80
$\mathcal{D}_1$	$BALD_{MCD}$	0.495 ± 0.038	0.497 ± 0.057	0.498 ± 0.057	0.486 ± 0.044	0.512 ± 0.057
	$BALD_{MCP}$	0.534 ± 0.018	0.584 ± 0.073	0.565 ± 0.033	0.619 ± 0.022	0.518 ± 0.028
	$BALC_{KLD}$	0.535 ± 0.038	0.521 ± 0.042	0.514 ± 0.053	0.511 ± 0.027	0.525 ± 0.037
	Temporal $BALC_{KLD}$	0.526 ± 0.040	0.538 ± 0.036	0.504 ± 0.028	0.501 ± 0.036	0.500 ± 0.004
$\mathcal{D}_2$	$BALD_{MCD}$	0.587 ± 0.044	0.564 ± 0.058	0.586 ± 0.047	0.613 ± 0.083	0.551 ± 0.031
	$BALD_{MCP}$	0.624 ± 0.044	0.598 ± 0.057	0.573 ± 0.053	0.560 ± 0.081	0.530 ± 0.015
	$BALC_{KLD}$	0.616 ± 0.043	0.653 ± 0.049	0.624 ± 0.051	0.565 ± 0.055	0.579 ± 0.019
	Temporal $BALC_{KLD}$	0.635 ± 0.045	0.603 ± 0.050	0.590 ± 0.042	0.602 ± 0.046	0.579 ± 0.041
$\mathcal{D}_3$	$BALD_{MCD}$	0.592 ± 0.015	0.604 ± 0.017	0.603 ± 0.016	0.603 ± 0.016	0.605 ± 0.018
	$BALD_{MCP}$	0.694 ± 0.047	0.730 ± 0.029	0.666 ± 0.034	0.639 ± 0.031	0.599 ± 0.035
	$BALC_{KLD}$	0.631 ± 0.006	0.631 ± 0.009	0.622 ± 0.011	0.631 ± 0.025	0.564 ± 0.046
	Temporal $BALC_{KLD}$	0.602 ± 0.011	0.622 ± 0.018	0.630 ± 0.014	0.618 ± 0.040	0.565 ± 0.050
$\mathcal{D}_4$	$BALD_{MCD}$	0.472 ± 0.029	0.472 ± 0.030	0.486 ± 0.008	0.476 ± 0.038	0.481 ± 0.038
	$BALD_{MCP}$	0.511 ± 0.021	0.510 ± 0.023	0.525 ± 0.033	0.498 ± 0.041	0.497 ± 0.017
	$BALC_{KLD}$	0.468 ± 0.022	0.472 ± 0.029	0.477 ± 0.029	0.483 ± 0.018	0.475 ± 0.032
	Temporal $BALC_{KLD}$	0.482 ± 0.023	0.491 ± 0.013	0.490 ± 0.021	0.487 ± 0.031	0.515 ± 0.022

J.2. Label Noise - Nearest Neighbour

Nearest neighbour noise is more realistic than that which is random as it may simulate uncertainty in diagnoses made by physicians. In this section, we illustrate, in Tables 9a - 9c, the degree to which the test AUC is affected by the introduction of nearest neighbour label noise during the active learning procedure.

As expected, extreme levels of noise negatively affect performance. For instance, this can be seen in Table 9a at $\mathcal{D}_3$ using $BALD_{MCD}$ where increasing the level of nearest neighbour noise from $5\% \rightarrow 80\%$ leads to a reduction of the $AUC = 0.744 \rightarrow 0.694$. SoQal continues to outperform Epsilon Greedy and Entropy Response across most of the noise levels. Building on the previous example, with 80% nearest neighbour noise, SoQal achieves an $AUC = 0.694$ whereas Epsilon Greedy and Entropy Response achieve an $AUC = 0.632$ and 0.587, respectively. Such a finding is similar to that arrived at with Random Noise and implies that SoQal is relatively more robust to noisy oracles than these other methods.

Table 9: Mean test AUC of oracle questioning strategies as a function of increasing levels of *nearest neighbour* label noise by the oracle. Results are shown for datasets $\mathcal{D}_1 - \mathcal{D}_5$ and all acquisition functions. Mean and standard deviation values are shown across five seeds.

(a) SoQal

Dataset	Ac. Function α	Nearest Neighbour Noise Level				
		0.05	0.10	0.20	0.40	0.80
$\mathcal{D}_1$	$BALD_{MCD}$	0.614 ± 0.043	0.571 ± 0.037	0.618 ± 0.015	0.557 ± 0.042	0.540 ± 0.052
	$BALD_{MCP}$	0.633 ± 0.011	0.617 ± 0.095	0.641 ± 0.023	0.632 ± 0.026	0.591 ± 0.047
	$BALC_{KLD}$	0.628 ± 0.049	0.586 ± 0.032	0.616 ± 0.024	0.604 ± 0.022	0.558 ± 0.069
	Temporal $BALC_{KLD}$	0.557 ± 0.045	0.647 ± 0.060	0.620 ± 0.036	0.625 ± 0.038	0.577 ± 0.039
$\mathcal{D}_2$	$BALD_{MCD}$	0.694 ± 0.022	0.631 ± 0.020	0.682 ± 0.036	0.658 ± 0.038	0.647 ± 0.039
	$BALD_{MCP}$	0.605 ± 0.054	0.660 ± 0.067	0.656 ± 0.029	0.618 ± 0.058	0.605 ± 0.081
	$BALC_{KLD}$	0.655 ± 0.015	0.660 ± 0.037	0.671 ± 0.078	0.649 ± 0.024	0.678 ± 0.023
	Temporal $BALC_{KLD}$	0.702 ± 0.044	0.654 ± 0.024	0.686 ± 0.038	0.638 ± 0.042	0.631 ± 0.020
$\mathcal{D}_3$	$BALD_{MCD}$	0.744 ± 0.023	0.745 ± 0.021	0.709 ± 0.028	0.700 ± 0.026	0.694 ± 0.014
	$BALD_{MCP}$	0.706 ± 0.029	0.736 ± 0.036	0.727 ± 0.023	0.712 ± 0.018	0.682 ± 0.017
	$BALC_{KLD}$	0.718 ± 0.029	0.729 ± 0.028	0.735 ± 0.021	0.680 ± 0.050	0.688 ± 0.009
	Temporal $BALC_{KLD}$	0.727 ± 0.033	0.725 ± 0.033	0.724 ± 0.018	0.700 ± 0.022	0.645 ± 0.062
$\mathcal{D}_4$	$BALD_{MCD}$	0.517 ± 0.034	0.477 ± 0.027	0.493 ± 0.034	0.498 ± 0.036	0.459 ± 0.035
	$BALD_{MCP}$	0.492 ± 0.027	0.491 ± 0.027	0.502 ± 0.036	0.532 ± 0.040	0.507 ± 0.042
	$BALC_{KLD}$	0.494 ± 0.024	0.494 ± 0.016	0.504 ± 0.026	0.506 ± 0.031	0.503 ± 0.021
	Temporal $BALC_{KLD}$	0.504 ± 0.018	0.515 ± 0.013	0.529 ± 0.027	0.507 ± 0.014	0.508 ± 0.026

(b) Epsilon Greedy

Dataset	Ac. Function α	Nearest Neighbour Noise Level				
		0.05	0.10	0.20	0.40	0.80
$\mathcal{D}_1$	$BALD_{MCD}$	0.503 ± 0.040	0.480 ± 0.023	0.514 ± 0.050	0.481 ± 0.038	0.456 ± 0.031
	$BALD_{MCP}$	0.501 ± 0.014	0.508 ± 0.036	0.522 ± 0.052	0.482 ± 0.023	0.496 ± 0.041
	$BALC_{KLD}$	0.541 ± 0.035	0.503 ± 0.033	0.486 ± 0.047	0.500 ± 0.041	0.473 ± 0.026
	Temporal $BALC_{KLD}$	0.516 ± 0.024	0.523 ± 0.044	0.495 ± 0.046	0.491 ± 0.013	0.483 ± 0.041
$\mathcal{D}_2$	$BALD_{MCD}$	0.584 ± 0.066	0.610 ± 0.042	0.597 ± 0.036	0.616 ± 0.054	0.593 ± 0.054
	$BALD_{MCP}$	0.565 ± 0.031	0.589 ± 0.075	0.616 ± 0.059	0.605 ± 0.047	0.586 ± 0.047
	$BALC_{KLD}$	0.608 ± 0.031	0.607 ± 0.040	0.590 ± 0.055	0.538 ± 0.037	0.585 ± 0.053
	Temporal $BALC_{KLD}$	0.647 ± 0.044	0.591 ± 0.033	0.640 ± 0.044	0.576 ± 0.031	0.589 ± 0.030
$\mathcal{D}_3$	$BALD_{MCD}$	0.656 ± 0.021	0.655 ± 0.014	0.665 ± 0.010	0.643 ± 0.021	0.632 ± 0.010
	$BALD_{MCP}$	0.660 ± 0.022	0.657 ± 0.023	0.659 ± 0.003	0.664 ± 0.023	0.634 ± 0.013
	$BALC_{KLD}$	0.608 ± 0.031	0.607 ± 0.040	0.590 ± 0.055	0.538 ± 0.037	0.585 ± 0.053
	Temporal $BALC_{KLD}$	0.644 ± 0.016	0.651 ± 0.011	0.658 ± 0.013	0.634 ± 0.016	0.627 ± 0.014
$\mathcal{D}_4$	$BALD_{MCD}$	0.438 ± 0.014	0.457 ± 0.022	0.442 ± 0.018	0.456 ± 0.028	0.428 ± 0.024
	$BALD_{MCP}$	0.489 ± 0.018	0.489 ± 0.021	0.474 ± 0.023	0.485 ± 0.019	0.486 ± 0.015
	$BALC_{KLD}$	0.485 ± 0.029	0.487 ± 0.019	0.495 ± 0.023	0.488 ± 0.028	0.481 ± 0.021
	Temporal $BALC_{KLD}$	0.486 ± 0.018	0.500 ± 0.029	0.486 ± 0.027	0.468 ± 0.017	0.487 ± 0.028

(c) Entropy Response

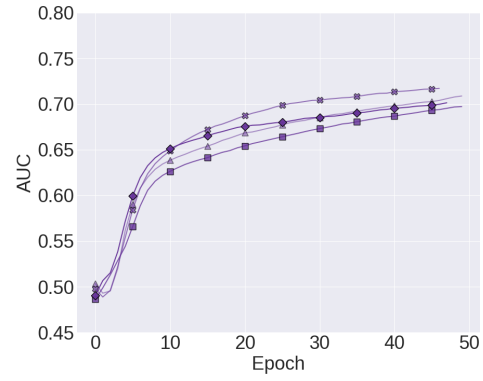| Dataset	Ac. Function α	Nearest Neighbour Noise Level				
		0.05	0.10	0.20	0.40	0.80
$\mathcal{D}_1$	$BALD_{MCD}$	0.494 ± 0.037	0.474 ± 0.027	0.492 ± 0.051	0.482 ± 0.033	0.444 ± 0.006
	$BALD_{MCP}$	0.511 ± 0.019	0.562 ± 0.052	0.509 ± 0.042	0.572 ± 0.060	0.495 ± 0.041
	$BALC_{KLD}$	0.513 ± 0.020	0.517 ± 0.035	0.504 ± 0.034	0.498 ± 0.023	0.487 ± 0.023
	Temporal $BALC_{KLD}$	0.500 ± 0.043	0.540 ± 0.025	0.503 ± 0.043	0.516 ± 0.024	0.490 ± 0.024
$\mathcal{D}_2$	$BALD_{MCD}$	0.585 ± 0.045	0.630 ± 0.056	0.600 ± 0.045	0.585 ± 0.046	0.586 ± 0.063
	$BALD_{MCP}$	0.633 ± 0.060	0.626 ± 0.064	0.618 ± 0.055	0.647 ± 0.077	0.619 ± 0.055
	$BALC_{KLD}$	0.605 ± 0.049	0.572 ± 0.032	0.630 ± 0.081	0.581 ± 0.031	0.589 ± 0.061
	Temporal $BALC_{KLD}$	0.625 ± 0.030	0.599 ± 0.024	0.613 ± 0.050	0.614 ± 0.052	0.606 ± 0.054
$\mathcal{D}_3$	$BALD_{MCD}$	0.604 ± 0.017	0.589 ± 0.013	0.592 ± 0.014	0.592 ± 0.014	0.587 ± 0.012
	$BALD_{MCP}$	0.636 ± 0.030	0.635 ± 0.030	0.640 ± 0.040	0.634 ± 0.039	0.623 ± 0.032
	$BALC_{KLD}$	0.632 ± 0.008	0.633 ± 0.008	0.630 ± 0.005	0.629 ± 0.004	0.625 ± 0.008
	Temporal $BALC_{KLD}$	0.631 ± 0.013	0.630 ± 0.013	0.637 ± 0.013	0.630 ± 0.014	0.629 ± 0.009
$\mathcal{D}_4$	$BALD_{MCD}$	0.475 ± 0.035	0.493 ± 0.025	0.471 ± 0.031	0.468 ± 0.027	0.481 ± 0.035
	$BALD_{MCP}$	0.508 ± 0.024	0.512 ± 0.020	0.513 ± 0.019	0.499 ± 0.012	0.492 ± 0.016
	$BALC_{KLD}$	0.483 ± 0.031	0.476 ± 0.033	0.473 ± 0.026	0.479 ± 0.021	0.479 ± 0.032
	Temporal $BALC_{KLD}$	0.490 ± 0.012	0.497 ± 0.030	0.466 ± 0.013	0.485 ± 0.016	0.500 ± 0.013

K. Effect of Number of Monte Carlo Samples, *T*, on Performance

The number of MC samples, T, within an AL framework can be associated with an improved approximation of the version space. This, in turn, should lead to improved AL results. To quantify the effect of the number of MC samples on performance, we illustrate in Fig. 14, the validation AUC for experiments conducted with $T = (5, 20, 40, 100)$. We show that there does not exist a simple proportional relationship between the number of MC samples and performance. This can be seen by the relatively strong generalization performance of models when $T = 100$ in Figs. 14c, 14h, and 14i and poorer performance when $T = 100$. This suggests that our family of methods can perform well without being computationally expensive.
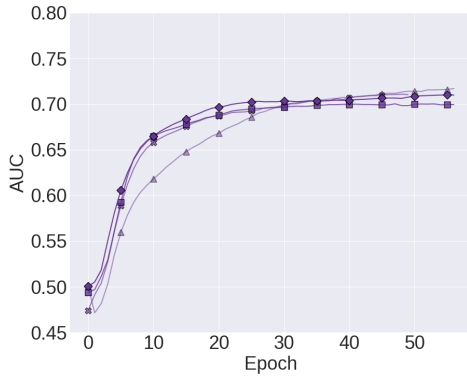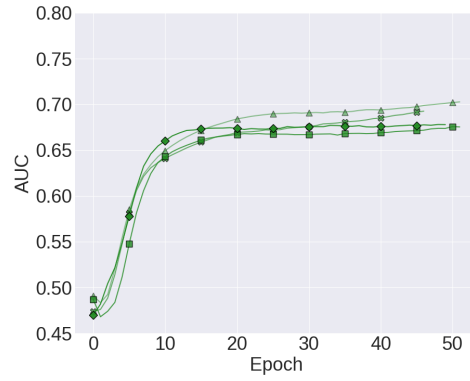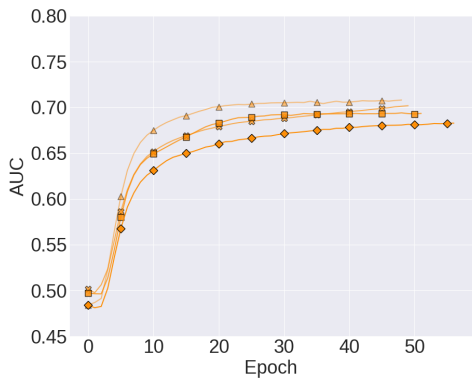
(a) Var Ratio

(b) Temporal Var Ratio

(c) Entropy

(d) Temporal Entropy

(e) BALD

(f) Temporal BALD

(g) BALC$_{\text{KLD}}$

(h) Temporal BALC$_{\text{KLD}}$

(i) BALC$_{\text{JSD}}$
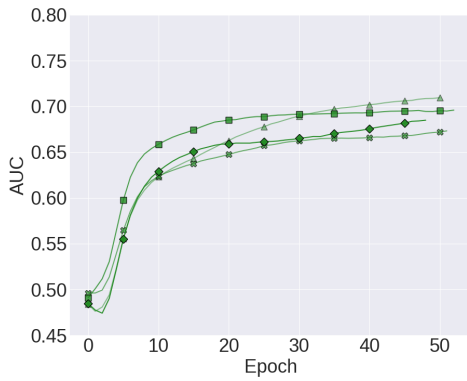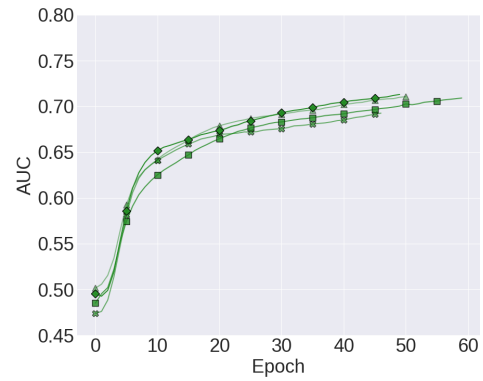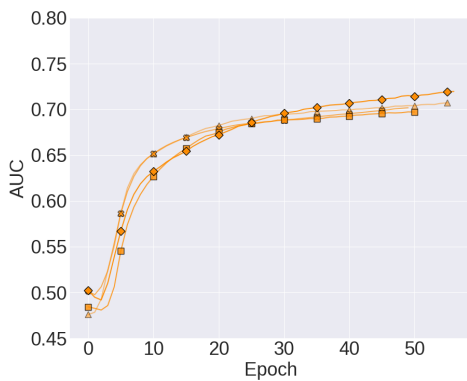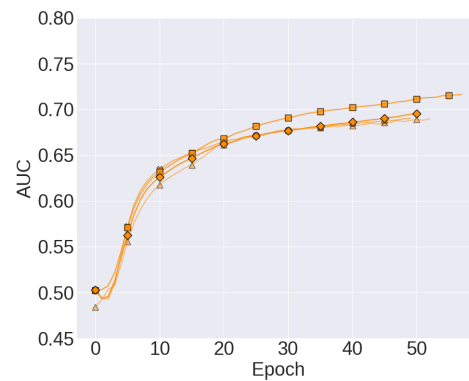
(j) Temporal BALC$_{\text{JSD}}$
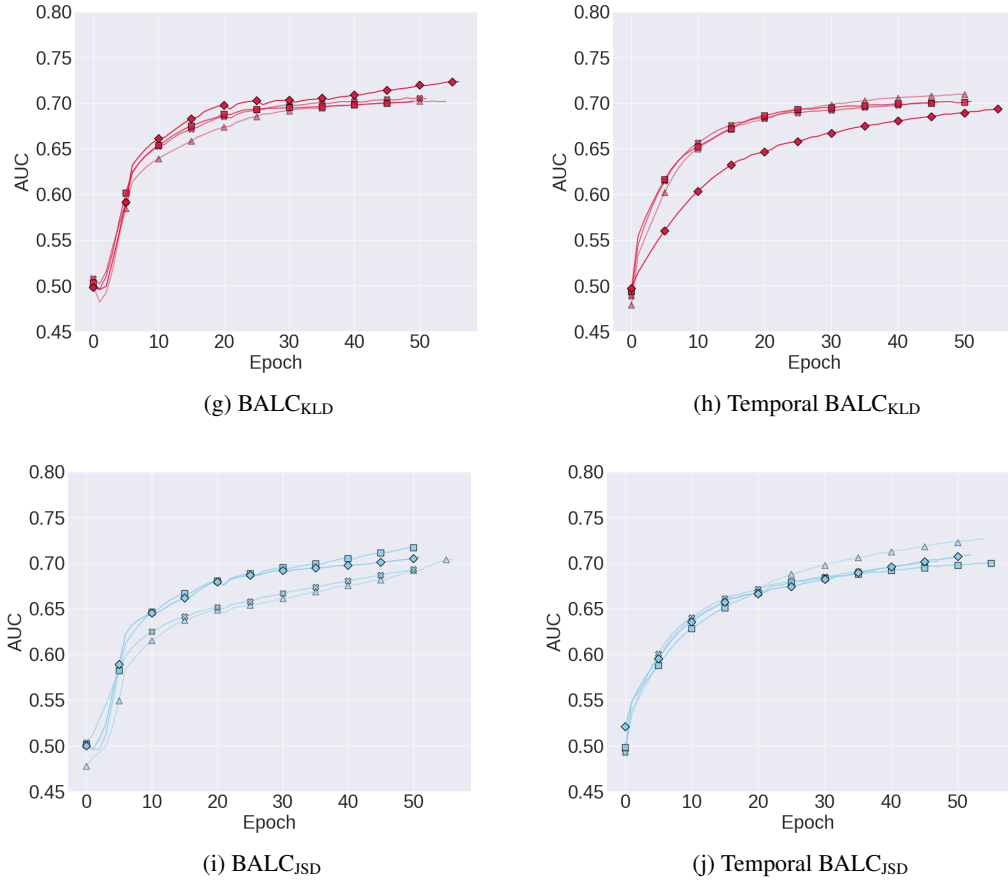
Figure 14: Mean validation AUC as a function of number of Monte Carlo samples T for the different acquisition functions using the MCP method. The acquisition percentage and acquisition epochs were fixed at $b = 2\%$ and $\tau = 5$, respectively. These experiments are performed on $\mathcal{D}_2$ at a fraction of $\beta = 0.5$. Results are averaged across 5 seeds.

L. Effect of Acquisition Percentage, *b*, on Performance

The number of unlabelled instances acquired during the AL procedure can have a strong effect on the generalization performance of networks. We investigate the effect of this on our family of methods and illustrate the results in Fig. 15 when conducting experiments for $b = (1\%, 2\%, 5\%, 20\%)$. Contrary to expectations that more acquisition is better, we show that acquiring large amounts of data is actually detrimental. This can be seen by the poorer performance attributed to $b = 20\%$ in, for instance, Figs. 15b, 15f, and 15g. We hypothesize that this is due to larger magnitude 1) distribution shifts and 2) label noise brought about by the absence of an oracle.

(a) Var Ratio

(b) Temporal Var Ratio

(c) Entropy

(d) Temporal Entropy

(e) BALD

(f) Temporal BALD

(g) BALC$_{\text{KLD}}$

(h) Temporal BALC$_{\text{KLD}}$

(i) BALC$_{\text{JSD}}$
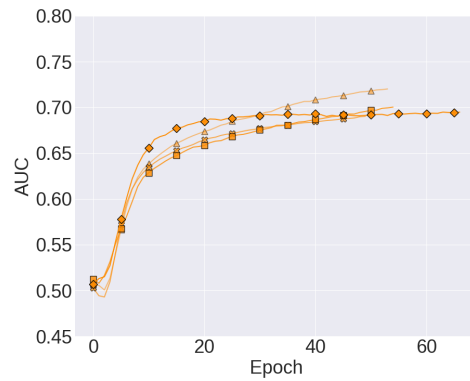
(j) Temporal BALC$_{\text{JSD}}$
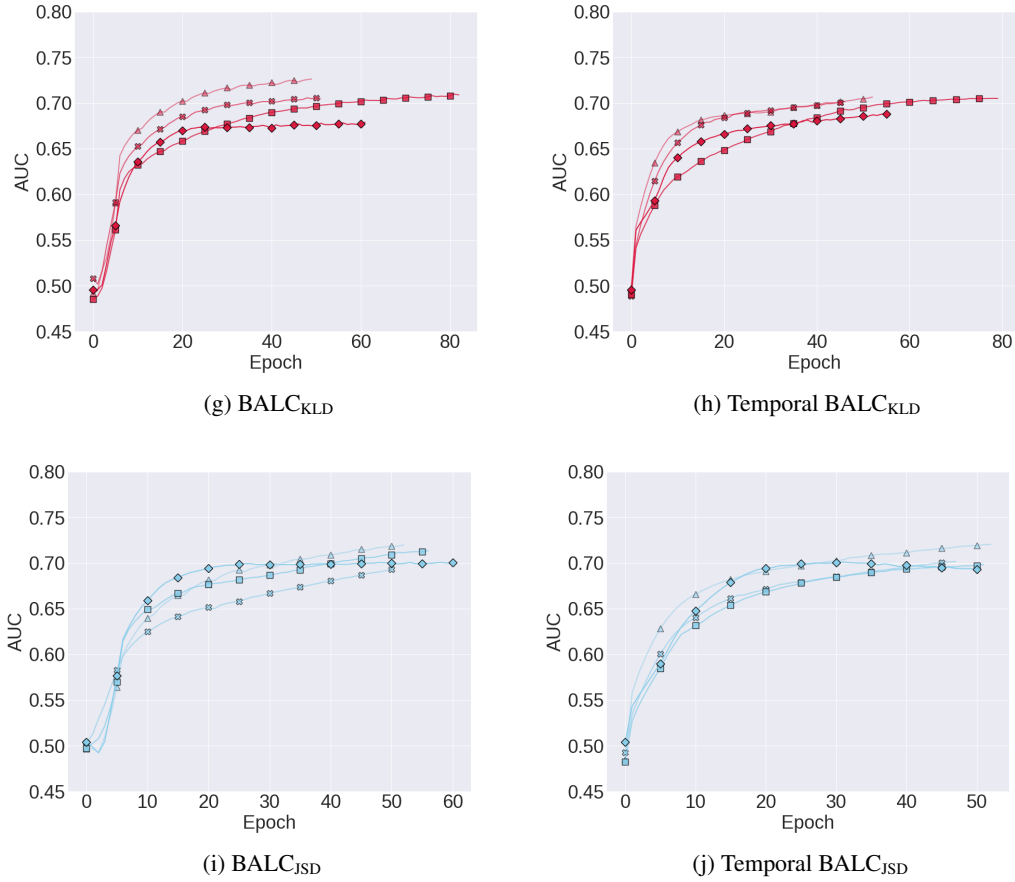
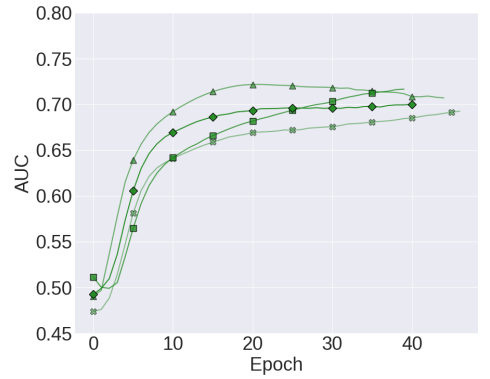Figure 15: Mean AUC of the validation set as a function of acquisition percentage b for the different acquisition functions using the MCP method. These experiments are performed on $\mathcal{D}_2$ at a fraction $\beta = 0.5$. MC samples and acquisition epochs were fixed at $T = 20$ and $\tau = 5$, respectively. Results are averaged across 5 seeds.

M. Effect of Acquisition Epochs, τ, on Performance

As outlined in the main manuscript, the control vs. shock trade-off must be balanced to ensure good generalization performance of an AL procedure. Acquiring instances too early and frequently can lead to instabilities in the training procedure. Conversely, inadequate sampling of unlabelled instances starves the network of much needed data. To quantify this trade-off, we illustrate in Fig. 16, the performance of our family of methods when $\tau = (5, 10, 15, 20)$. Although one value that guarantees best performance for all experiments does not exist, $\tau = 10$ or $\tau = 15$ seem to outperform the others, on average.
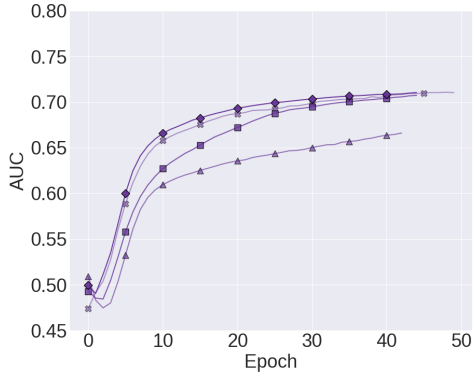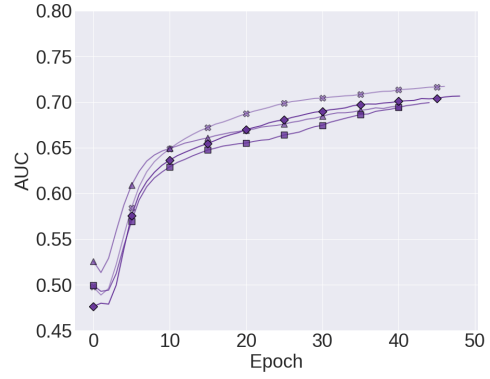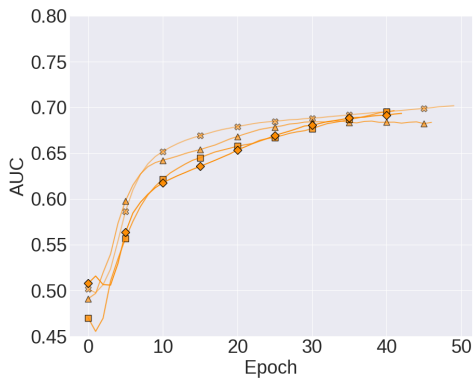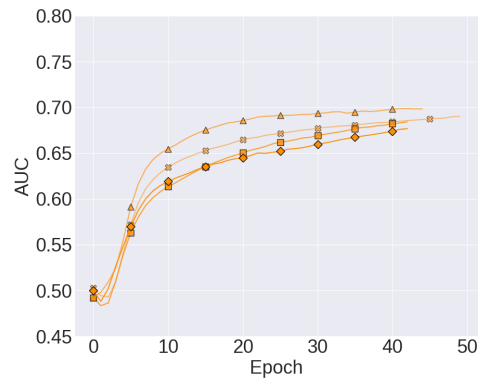
(a) Var Ratio

(b) Temporal Var Ratio
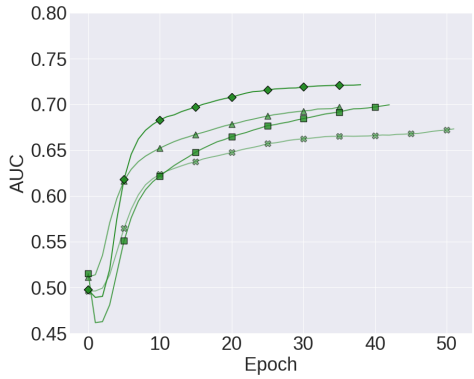
(c) Entropy

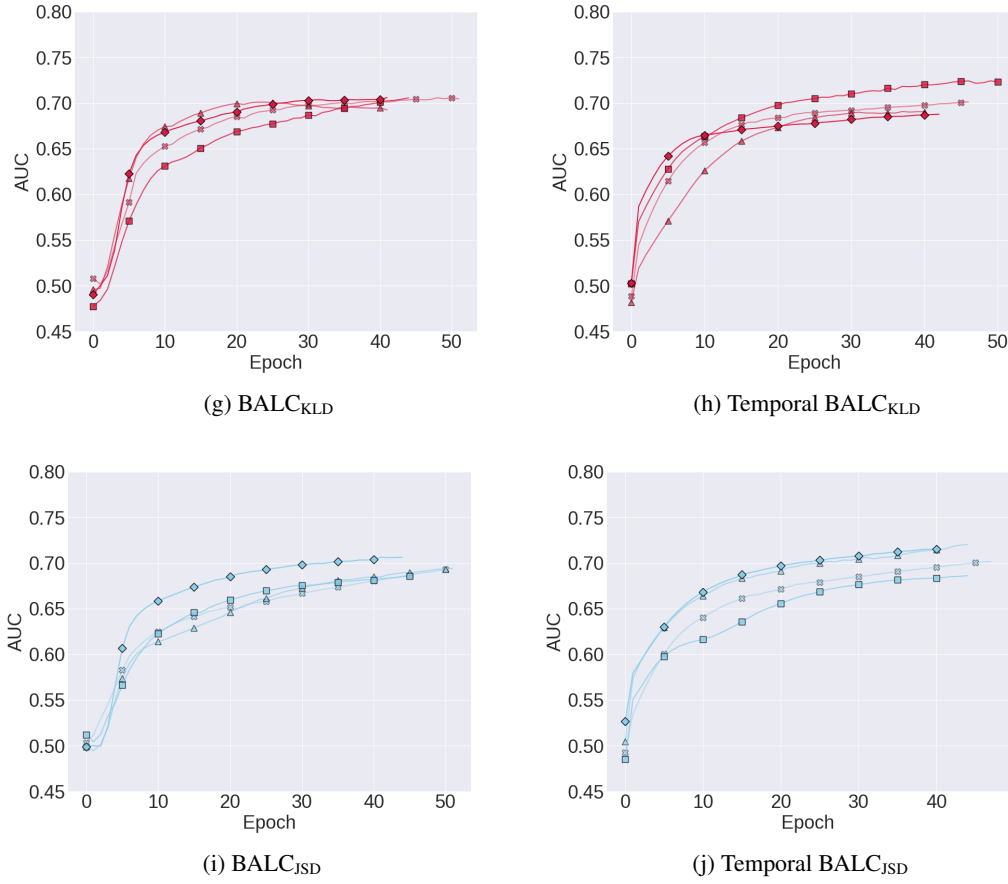(d) Temporal Entropy

(e) BALD

(f) Temporal BALD

Figure 16: Mean AUC of the validation set as a function of acquisition epochs τ for the different acquisition functions using the MCP method. MC samples and the acquistion percentage were fixed at $T = 20$ and $b = 2\%$, respectively. These experiments are performed on $\mathcal{D}_2$ at a fraction $\beta = 0.5$. Results are averaged across 5 seeds.