CLOPS: CONTINUAL LEARNING OF PHYSIOLOGICAL SIGNALS

Dani Kiyasseh
Department of Engineering Science
University of Oxford
Oxford, UK
dani.kiyasseh@eng.ox.ac.uk

Tingting Zhu
Department of Engineering Science
University of Oxford
Oxford, UK
tingting.zhu@eng.ox.ac.uk

David A. Clifton
Department of Engineering Science
University of Oxford
Oxford, UK
david.clifton@eng.ox.ac.uk

ABSTRACT

Deep learning algorithms are known to experience destructive interference when instances violate the assumption of being independent and identically distributed (i.i.d). This violation, however, is ubiquitous in clinical settings where data are streamed temporally and from a multitude of physiological sensors. To overcome this obstacle, we propose CLOPS, a replay-based continual learning strategy. In three continual learning scenarios based on three publically-available datasets, we show that CLOPS can outperform the state-of-the-art methods, GEM and MIR. Moreover, we propose end-to-end trainable parameters, which we term task-instance parameters, that can be used to quantify task difficulty and similarity. This quantification yields insights into both network interpretability and clinical applications, where task difficulty is poorly quantified.

1 INTRODUCTION

Many deep learning algorithms operate under the assumption that instances are independent and identically-distributed (i.i.d.). The violation of this assumption can be detrimental to the training behaviour and performance of an algorithm. The assumption of independence can be violated, for example, when data are streamed temporally from a sensor. Introducing multiple sensors in a changing environment can introduce covariate shift, arguably the 'Achilles heel' of machine learning model deployment (Quionero-Candela et al., 2009).

A plethora of realistic scenarios violate the i.i.d. assumption. This is particularly true in healthcare where the multitude of physiological sensors generate time-series recordings that may vary temporally (due to seasonal diseases; e.g. flu), across patients (due to different hospitals or hospital settings), and in their modality. Tackling the challenges posed by such scenarios is the focus of continual learning (CL) whereby a learner, when exposed to tasks in a sequential manner, is expected to perform well on current tasks without compromising performance on previously seen tasks. The outcome is a single algorithm that can reliably solve a multitude of tasks. However, most, if not all, research in this field has been limited to a small handful of imaging datasets (Lopez-Paz & Ranzato, 2017; Aljundi et al., 2019b;a). Although understandable from a benchmarking perspective, such research fails to explore the utility of continual learning methodologies in more realistic healthcare scenarios (Farquhar & Gal, 2018). To the best of our knowledge, we are the first to explore and propose a CL approach in the context of physiological signals. The dynamic and chaotic environment that characterizes healthcare necessitates the availability of algorithms that are dynamically reliable; those that can adapt to potential covariate shift without catastrophically forgetting how to perform tasks from the past. Such dynamic reliability implies that algorithms no longer needs to be retrained on data or tasks to which it has been exposed in the past, thus improving its data-efficiency. Secondly, algorithms that

perform consistently well across a multitude of tasks are more trustworthy, a desirable trait sought by medical professionals (Spiegelhalter, 2020).

Our Contributions. In this paper, we propose a replay-based continual learning methodology that is based on the following:

1. **Importance-guided storage:** task-instance parameters, a scalar corresponding to each instance in each task, as informative signals for *loss-weighting* and *buffer-storage*.
2. **Uncertainty-based acquisition:** an active learning inspired methodology that determines the degree of informativeness of an instance and thus acts as a *buffer-acquisition* mechanism.

2 Related Work

Continual learning (CL) approaches have resurfaced in recent years (Parisi et al., 2019). Those similar to ours comprise memory-based methods such as iCaRL (Rebuffi et al., 2017), CLEAR (Rolnick et al., 2019), GEM (Lopez-Paz & Ranzato, 2017), and aGEM (Chaudhry et al., 2018). In contrast to our work, the latter two methods naively populate their replay buffer with the last m examples observed for a particular task. Isele & Cosgun (2018) and Aljundi et al. (2019b) employ a more sophisticated buffer-storage strategy where a quadratic programming problem is solved in the absence of task boundaries. Aljundi et al. (2019a) introduce MIR whereby instances are stored using reservoir sampling and sampled according to whether they incur the greatest change in loss if parameters were to be updated on the subsequent task. This approach is computationally expensive, requiring multiple forward and backward passes per batch. The application of CL in the medical domain is limited to that of Lenga et al. (2020) wherein existing methodologies are implemented on chest X-ray datasets. In contrast to previous research that independently investigates buffer-storage and acquisition strategies, we focus on a *dual* storage and acquisition strategy.

Active learning (AL) in healthcare has observed increased interest in recent years, with a review of methodologies provided by Settles (2009). For example, Gong et al. (2019) propose a Bayesian deep latent Gaussian model to acquire important features from electronic health record (EHR) data in MIMIC (Johnson et al., 2016) to improve mortality prediction. In dealing with EHR data, Chen et al. (2013) use the distance of unlabelled samples from the hyperplane in an SVM to acquire datapoints. Wang et al. (2019) implement an RNN to acquire ECG samples during training. Zhou et al. (2017) perform transfer learning in conjunction with a convolutional neural network to acquire biomedical images in an online manner. Smailagic et al. (2018; 2019) actively acquire unannotated medical images by measuring their distance in a latent space to images in the training set. Such similarity metrics, however, are sensitive to the amount of available labelled training data. Gal et al. (2017) adopt BALD (Houlsby et al., 2011) with Monte Carlo Dropout to acquire instances that maximize the Jensen-Shannon divergence (JSD) across MC samples. To the best of our knowledge, we are the first to employ AL-inspired acquisition functions in the context of CL.

3 Background

3.1 Continual Learning

In this work, we consider a learner, $f_\omega : x_\mathcal{T} \in \mathbb{R}^m \rightarrow y_\mathcal{T} \in \mathbb{R}^c$, parameterized by ω, that maps an m-dimensional input, $x_\mathcal{T}$, to a c-dimensional output, $y_\mathcal{T}$, where c is the number of classes, for each task $\mathcal{T} \in [1 \dots N]$. This learner is exposed to new tasks in a sequential manner once previously-tackled tasks are mastered. In this paper, we formulate our tasks based on a modification of the three-tier categorization proposed by van de Ven & Tolias (2019). In our learning scenarios (see Fig. 1), a network is sequentially tasked with solving a binary classification problem in response to data from mutually-exclusive pairs of classes **Class Incremental Learning (Class-IL)**, multi-class classification problem in response to data collected at different times of the year (e.g., winter and summer) **Time Incremental Learning (Time-IL)**, and a multi-class classification problem in response to inputs with a different modality **Domain Incremental Learning (Domain-IL)**. In the aforementioned cases, task identities are *absent* during both training and testing and neural architectures are single-headed.

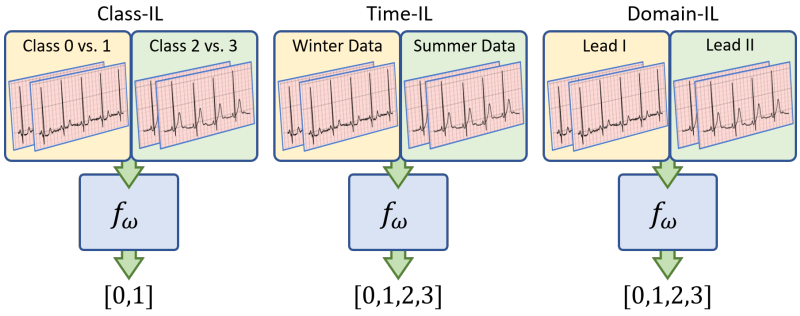


Figure 1: Illustration of the three continual learning scenarios. A network is sequentially exposed to tasks **(Class-IL)** with mutually-exclusive pairs of classes, **(Time-IL)** with data collected at different times of the year, and **(Domain-IL)** with data from different input modalities.

4 METHODS

The two ideas underlying our proposal are the storage of instances into *and* the acquisition of instances from a buffer such that destructive interference is mitigated. We describe these in more detail below.

4.1 IMPORTANCE-GUIDED BUFFER STORAGE

We aim to populate a buffer, $\mathcal{D}_B$, of finite size, $\mathcal{M}$, with instances from the current task that are considered important. To quantify importance, we learn parameters, entitled task-instance parameters, $\beta_{i\mathcal{T}}$, associated with each instance, $x_{i\mathcal{T}}$, in each task, $\mathcal{T}$. These parameters play a dual role.

4.1.1 LOSS-WEIGHTING MECHANISM

For the current task, k, and its associated data, $\mathcal{D}_k$, we incorporate β as a coefficient of the loss, $\mathcal{L}_{ik}$, incurred for each instance, $x_{ik} \in \mathcal{D}_k$. For a mini-batch of size, B, that consists of B_k instances from the current task, the objective function is shown in Eq. 1. We can learn the values of β_{ik} via gradient descent, with some learning rate, η, as shown in Eq. 2.

$$\mathcal{L} = \frac{1}{B_k}\sum_{i=1}^{B_k} \beta_{ik}\mathcal{L}_{ik} \qquad (1) \qquad\qquad \beta_{ik} \leftarrow \beta_{ik} - \eta\frac{\partial \mathcal{L}}{\partial \beta_{ik}} \qquad (2)$$

Note that $\frac{\partial \mathcal{L}}{\partial \beta_{ik}} = \mathcal{L}_{ik} > 0$. This suggests that instances that are hard to classify ($\uparrow \mathcal{L}_{ik}$) will exhibit $\downarrow \beta_{ik}$. From this perspective, β_{ik} can be viewed as a proxy for instance difficulty. However, as presented, $\beta_{ik} \to 0$ as training progresses, an observation we confirmed empirically. Since β_{ik} is the coefficient of the loss, $\mathcal{L}_{ik}$, this implies that the network will quickly be unable to learn from the data. To avoid this behaviour, we initialize $\beta_{ik} = 1$ in order to emulate a standard loss function and introduce a regularization term to penalize its undesirable and rapid decay toward zero. As a result, our modified objective function is:

$$\mathcal{L}_{\text{current}} = \frac{1}{B_k}\sum_{i=1}^{B_k} \beta_{ik}\mathcal{L}_{ik} + \lambda(\beta_{ik} - 1)^2 \qquad (3)$$

When $k > 1$, we replay instances from previous tasks by using a replay buffer (see Sec. 4.2 for replay mechanism). These replayed instances incur a loss $\mathcal{L}_{ij}\ \forall\ j \in [1 \ldots k-1]$. We decide to not weight these instances, in contrast to what we perform to instances from the current task (see Appendix K).

$$\mathcal{L}_{\text{replay}} = \frac{1}{B - B_k}\sum_{j=1}^{k-1}\sum_{i}^{B_j} \mathcal{L}_{ij} \qquad (4) \qquad\qquad \mathcal{L} = \mathcal{L}_{\text{current}} + \mathcal{L}_{\text{replay}} \qquad (5)$$

4.1.2 BUFFER-STORAGE MECHANISM

We leverage β, as a proxy for instance difficulty, to store instances into the buffer. To describe the intuition behind this process, we illustrate, in Fig. 2, the trajectory of β_{1k} and β_{2k} associated with two instances, x_{1k} and x_{2k}, while training on the current task, k, for $\tau = 20$ epochs. In selecting instances for storage into the buffer, we can 1) retrieve their corresponding β values at the *conclusion* of the task, i.e., at $\beta(t = 20)$, 2) rank all instances based on these β values, and 3) acquire the top b fraction of instances. This approach, however, can lead to *erroneous* estimates of the relative difficulty of instances, as explained next.

In Fig. 2, we see that $\beta_{2k} > \beta_{1k}$ for the majority of the training process, indicating that x_{2k} had been easier to classify than x_{1k}. The swap in the ranking of these β values that occurs towards the end of training in addition to myopically looking at $\beta(t = 20)$ would *erroneously* make us believe that the opposite was true. Such convergence or swapping of β values has also been observed by Saxena et al. (2019). As a result, the reliability of β as a proxy of instance difficulty is eroded.

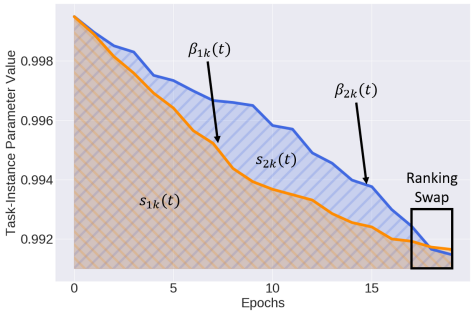


Figure 2: Trajectory of β_{1k} and β_{2k} on task k. Ranking instances based on $\beta(t = 20)$ leads to erroneous estimates of their relative difficulty. We propose to rank instances based on the area under the trajectory of β, denoted as s_{ik}.

To maintain the reliability of this proxy, we propose to *track* the β values after each training epoch, t, until the final epoch, τ, for the task at hand and calculate the area under these tracked values. We do so by using the trapezoidal rule as shown in Eq. 6. We explored several variants of the storage function and found the proposed form to work best (see Appendix H). At $t = \tau$, we rank the instances in descending order of s_{ik} (easy to hard) as we found this preferable to the opposite order (see Appendix I), select the top b fraction, and store them into the buffer, of which each task is allotted a fixed portion. The higher the value of the storage fraction, b, the more likely it is that the buffer will contain representative instances and thus mitigate forgetting, however this comes at an increased computational cost.

$$s_{ik} = \int_0^{\tau} \beta_{ik}(t)dt \approx \sum_{t=0}^{\tau} \left(\frac{\beta_{ik}(t + \Delta t) + \beta_{ik}(t)}{2} \right) \Delta t \tag{6}$$

4.2 UNCERTAINTY-BASED BUFFER ACQUISITION

The acquisition of instances that a learner is uncertain about is likely to benefit training (Zhu, 2005). This is the premise of uncertainty-based acquisition functions such as BALD (Houlsby et al., 2011; Gal & Ghahramani, 2016). We now outline how to exploit this premise for buffer acquisition.

At epoch number, τ_{MC}, referred to as Monte Carlo (MC) epochs, each of the M instances, $x \sim \mathcal{D}_B$, is passed through the network and exposed to a stochastic binary dropout mask to generate an output, $p(y|x, \omega) \in \mathbb{R}^C$. This is repeated T times to form a matrix, $G \in \mathbb{R}^{M \text{x} T \text{x} C}$. An acquisition function, such as BALD_{MCD}, is thus a function $\mathcal{F} : \mathbb{R}^{M \text{x} T \text{x} C} \rightarrow \mathbb{R}^M$.

$$\text{BALD}_{\text{MCD}} = \text{JSD}(p_1, p_2, \ldots, p_T) = \text{H}\left(p(y|x)\right) - \mathbb{E}_{p(\omega|D_{train})}\left[\text{H}\left(p(y|x, \hat{\omega})\right)\right] \tag{7}$$

where $\text{H}(p(y|x))$ represents the entropy of the network outputs averaged across the MC samples, and $\hat{\omega} \sim p(\omega|D_{train})$ as in Gal & Ghahramani (2016). At sample epochs, τ_S, we rank instances in descending order of BALD_{MCD} and acquire the top a fraction from each task in the buffer. A higher value of this acquisition fraction, a, implies more instances are acquired. Although this may not guarantee improvement in performance, it does guarantee increased training overhead. Nonetheless, the intuition is that by acquiring instances, from previous tasks, to which a network is most confused, it can be nudged to avoid destructive interference in a data-efficient manner. We outline the entire training procedure in Algorithms 1-4 in Appendix A.

5 EXPERIMENTAL DESIGN

5.1 DATASETS

We conduct experiments[1] in PyTorch (Paszke et al., 2019). Given our emphasis on healthcare, we evaluate our approach on three publically-available datasets that include physiological time-series data such as the electrocardiogram (ECG) alongside cardiac arrhythmia labels. We use $\boldsymbol{\mathcal{D}_1}$ **= Cardiology ECG** (Hannun et al., 2019) (12-way), $\boldsymbol{\mathcal{D}_2}$ **= Chapman ECG** (Zheng et al., 2020) (4-way), and $\boldsymbol{\mathcal{D}_3}$ **= PhysioNet 2020 ECG** (Perez Alday et al., 2020) (9-way, multi-label). Further details regarding the datasets and network architecture can be found in Appendix C.

5.2 CONTINUAL LEARNING SCENARIOS

Here, we outline the three primary continual learning scenarios we use for our experiments. In **Class-IL**, $\mathcal{D}_1$ is split according to mutually-exclusive pairs of classes $[0, 1]$, $[2, 3]$, $[4, 5]$, $[6, 7]$, $[8, 9]$, and $[10, 11]$. This scenario allows us to evaluate the sensitivity of a network to new classes. In **Time-IL**, $\mathcal{D}_2$ is split into three tasks; Term 1, Term 2, and Term 3 corresponding to mutually-exclusive times of the year during which patient data were collected. This scenario allows us to evaluate the effect of temporal non-stationarity on a network's performance. Lastly, in **Domain-IL**, $\mathcal{D}_3$ is split according to the 12 leads of an ECG; 12 different projections of the same electrical signal generated by the heart. This scenario allows us to evaluate how robust a network is to the input distribution.

5.3 BASELINE METHODS

We compare our proposed method to the following. **Multi-Task Learning (MTL)** (Caruana, 1993) is a strategy whereby all datasets are assumed to be available at the same time and thus can be simultaneously used for training. Although this assumption may not hold in clinical settings due to the nature of data collection, privacy or memory constraints, it is nonetheless a strong baseline. **Fine-tuning** is a strategy that involves updating all parameters when training on subsequent tasks as they arrive without explicitly dealing with catastrophic forgetting. We also adapt two replay-based methods for our scenarios. **GEM** (Lopez-Paz & Ranzato, 2017) solves a quadratic programming problem to generate parameter gradients that do not increase the loss incurred by replayed instances. **MIR** (Aljundi et al., 2019a) replays instances from a buffer that incur the greatest change in loss given a parameter pseudo-update. Details on how these methods were adapted are found in Appendix C.

5.4 EVALUATION METRICS

To evaluate our methods, we exploit metrics suggested by Lopez-Paz & Ranzato (2017) such as average AUC and Backward Weight Transfer (BWT). We also propose two additional evaluation metrics that provide us with a more fine-grained analysis of learning strategies.

t-Step Backward Weight Transfer. To determine how performance changes 't-steps into the future', we propose BWT_t which evaluates the performance of the network on a previously-seen task, after having trained on t-tasks after it.

$$\mathrm{BWT}_t = \frac{1}{N-t} \sum_{j=1}^{N-t} \mathrm{R}_j^{j+t} - \mathrm{R}_j^j \tag{8}$$

Lambda Backward Weight Transfer. We extend BWT_t to all time-steps, *t*, to generate BWT_λ. As a result, we can identify improvements in methodology at the task-level.

$$\mathrm{BWT}_\lambda = \frac{1}{N-1} \sum_{j=1}^{N-1} \left[\frac{1}{N-j} \sum_{t=1}^{N-j} \mathrm{R}_j^{j+t} - \mathrm{R}_j^j \right] \tag{9}$$

5.5 HYPERPARAMETERS

Depending on the continual learning scenario, we chose $\tau = 20$ or 40, as we found that to achieve strong performance on the respective validation sets. We chose $\tau_{MC} = 40 + n$ and the sample epochs

[1] Our code is available at: https://github.com/danikiyasseh/CLOPS

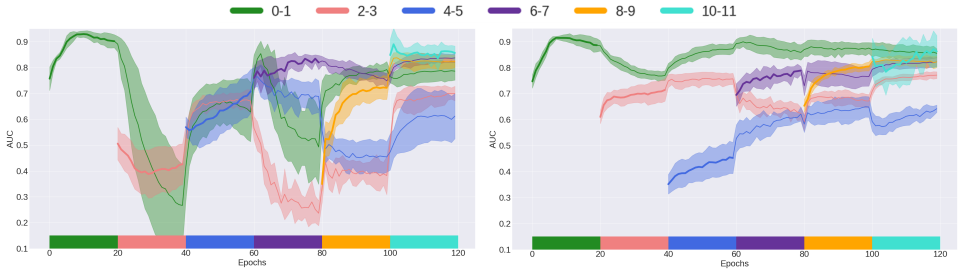
$\tau_S = 41 + n$ where $n \in \mathbb{N}^+$ in order to sample data from the buffer at every epoch following the first task. The values must satisfy $\tau_S \geq \tau_{MC} > \tau$. For computational reasons, we chose the storage fraction $b = 0.1$ of the size of the training dataset and the acquisition fraction $a = 0.25$ of the number of samples per task in the buffer. To calculate the acquisition function, we chose the number of Monte Carlo samples, $T = 20$. We chose the regularization coefficient, $\lambda = 10$. We also explore the effect of changing these values on performance (see Appendices L and M).

6 Experimental Results

6.1 Class-IL

Destructive interference is notorious amongst neural networks. In this section, we quantify such interference when learners are exposed to tasks involving novel classes. In Fig. 3a, we illustrate the AUC achieved on sequential binary classification tasks. We find that destructive interference is prevalent. For example, the network quickly forgets how to perform task $[0-1]$ once exposed to data from task $[2-3]$. This can be seen by the AUC $\approx 0.92 \rightarrow 0.30$. The final performance of the network for that particular task (AUC ≈ 0.78) is also lower than that maximally-achieved. In Fig. 3b, we show that CLOPS alleviates this interference. This can be seen by the absence of significant drops in AUC and higher final performance for all tasks relative to the fine-tuning strategy.

In Table 1, we compare the performance of the CL strategies in the Class-IL scenario. We find that CLOPS outperforms MTL (AUC $= 0.796$ vs. 0.701), which is a seemingly non-intuitive finding. We hypothesize that this finding is due to positive weight transfer brought about by a curriculum wherein sequential tasks of different levels of difficulty can improve generalization performance (Bengio et al., 2009). We explore this hypothesis further in Sec. 6.5. We also find that CLOPS outperforms state-of-the-art methods, GEM and MIR, in terms of generalization performance and exhibits constructive interference. For example, CLOPS and MIR achieve an AUC $= 0.796$ and 0.753, respectively. Moreover, BWT $= 0.053$ and 0.009 for these two methods, respectively. Such a finding underscores the ability of CLOPS to deal with tasks involving novel classes. We also show that CLOPS is robust to task order (see Appendix F).



(a) Fine-tuning (b) CLOPS

Figure 3: Mean validation AUC of the a) fine-tuning and b) CLOPS strategy ($b = 0.25$ and $a = 0.50$) in the Class-IL scenario. Coloured blocks indicate tasks on which the learner is currently being trained. The shaded area represents one standard deviation across five seeds.

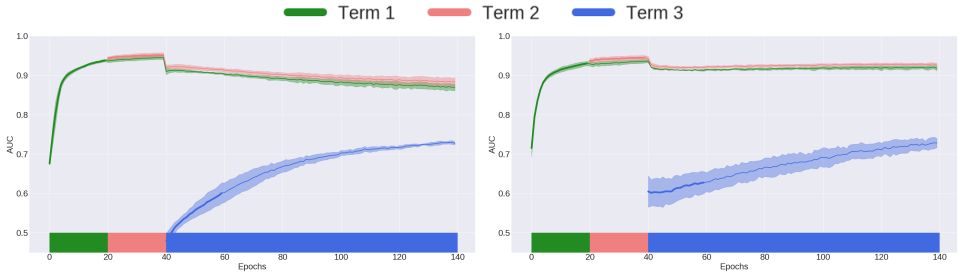
Table 1: Performance of CL strategies in the Class-IL scenario. Storage and acquisition fractions are $b = 0.25$ and $a = 0.50$, respectively. Mean and standard deviation are shown across five seeds.

Method	Average AUC	BWT	BWT_t	BWT_λ
MTL	0.701 ± 0.014	-	-	-
Fine-tuning	0.770 ± 0.020	0.037 ± 0.037	$(0.076) \pm 0.064$	$(0.176) \pm 0.080$
Replay-based Methods				
GEM	0.544 ± 0.031	$(0.024) \pm 0.028$	$(0.046) \pm 0.017$	$(0.175) \pm 0.021$
MIR	0.753 ± 0.014	0.009 ± 0.018	0.001 ± 0.025	$(0.046) \pm 0.022$
CLOPS	$\mathbf{0.796 \pm 0.013}$	$\mathbf{0.053 \pm 0.023}$	$\mathbf{0.018 \pm 0.010}$	$\mathbf{0.008 \pm 0.016}$

6.2 TIME-IL

Environmental changes within healthcare can introduce seasonal shift into datasets. In this section, we quantify the effect of such a shift on learners. In Fig. 4a, we illustrate the AUC achieved on tasks with seasonally-shifted data.

In this scenario, we find that CLOPS is capable of achieving forward weight transfer (FWT). For example, in Figs. 4a and 4b, CLOPS achieves an AUC ≈ 0.62 after one epoch of training on task Term 3, a value that the fine-tuning strategy only achieves after 20 epochs, signalling a 20-fold reduction in training time. We attribute this FWT to the loss-weighting role played by the task-instance parameters. By placing greater emphasis on more useful instances, the generalization performance of the network is improved. We also find that CLOPS exhibits reduced catastrophic forgetting relative to fine-tuning. For example, performance on tasks Term 1 and Term 2 is maintained at AUC > 0.90 when training on task Term 3. We do not observe this for the fine-tuning setup.



(a) Fine-tuning (b) CLOPS

Figure 4: Mean validation AUC of the (a) fine-tuning and (b) CLOPS strategy in the Time-IL scenario. Coloured blocks indicate tasks on which the learner is currently being trained. The shaded area represents one standard deviation from the mean across five seeds.

6.3 DOMAIN-IL

So far, we have shown the potential of CLOPS to alleviate destructive interference and allow for forward weight transfer. In this section, and in Table 2, we illustrate the performance of the CL strategies in the Domain-IL scenario. We show that CLOPS outperforms state-of-the-art methods. For example, CLOPS and MIR achieve an AUC $= 0.731$ and 0.716, respectively. CLOPS is also better at mitigating destructive interference, as shown by BWT $= (0.011)$ and (0.022), respectively. We provide an explanation for such performance by conducting ablation studies in the next section.

Table 2: Performance of CL strategies in the Domain-IL scenario. Storage and acquisition fractions are $b = 0.25$ and $a = 0.50$, respectively. Mean and standard deviation are shown across five seeds.

Method	Average AUC	BWT	BWT_t	BWT_λ
MTL	0.730 ± 0.016	-	-	-
Fine-tuning	0.687 ± 0.007	$(0.041) \pm 0.008$	$(0.047) \pm 0.004$	$(0.070) \pm 0.007$
Replay-based Methods				
GEM	0.502 ± 0.012	$(0.025) \pm 0.008$	$\mathbf{0.004 \pm 0.010}$	$(0.046) \pm 0.021$
MIR	0.716 ± 0.011	$(0.022) \pm 0.011$	$(0.013) \pm 0.004$	$(0.019) \pm 0.006$
CLOPS	$\mathbf{0.731 \pm 0.001}$	$\mathbf{(0.011) \pm 0.002}$	$(0.020) \pm 0.004$	$(0.019) \pm 0.009$

6.4 EFFECT OF TASK-INSTANCE PARAMETERS, β, AND ACQUISITION FUNCTION, α

To better understand the root cause of CLOPS' benefits, we conduct three ablation studies: 1) **Random Storage** dispenses with task-instance parameters and instead randomly stores instances into the buffer, 2) **Random Acquisition** dispenses with acquisition functions and instead randomly

acquires instances from the buffer, and 3) **Random Storage and Acquisition** which stores instances into, and acquires instances from, the buffer randomly. In Fig. 5, we illustrate the effect of these strategies on performance as we vary the storage fraction, b, and acquisition fraction, a.

We find that β, as a loss-weighting mechanism, benefits generalization performance. For example, in Fig. 5 (red rectangle), at $b = 1, a = 0.5$, we show that simply including the loss-weighting mechanism $\uparrow$ AUC $\approx 12\%$. We hypothesize that this mechanism is analogous to attention being placed on instance losses and thus allows the network to learn *which* instances to exploit further. We also find that uncertainty-based acquisition functions offer significant improvements. In Fig. 5 (black rectangles), at $a = 0.1, b = 0.5$, we show that such acquisition $\uparrow$ AUC $\approx 8\%$. We arrive at the same conclusion when evaluating backward weight transfer (see Appendix J).

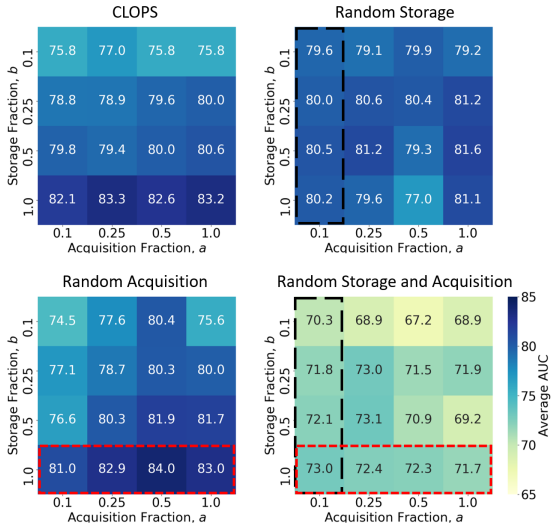


Figure 5: Mean validation AUC of four learning strategies in the Class-IL scenario. Results are shown as a function of storage fractions, b, and acquisition fractions, a and are an average across five seeds. We highlight the utility of (red rectangle) task-instance parameters as a loss-weighting mechanism and (black rectangle) uncertainty-based acquisition functions for acquiring instances from the buffer.

6.5 Validation of Interpretation of Task-Instance Parameters, β

We claimed that instances with lower values of β, and by extension, s, are relatively more difficult to classify. In this section, we aim to validate this intuition. In Fig. 6, we illustrate the distribution of s values corresponding to each task.

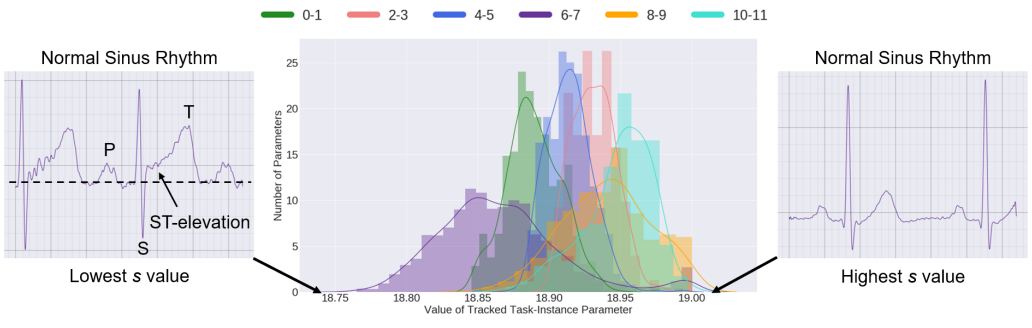


Figure 6: Distribution of the s values corresponding to CLOPS ($b = 0.25$ and $a = 0.50$) in the Class-IL scenario. Each colour corresponds to a different task. The ECG recording with the lowest s value is labelled as normal despite the presence of ST-elevation, a feature common in heart attacks.

We find that tasks differ in their difficulty level. For example, task $[6-7]$ is considered more difficult to solve than task $[8-9]$ as evidenced by the lower distribution mean of the former relative to the latter ($s \approx 18.85$ vs. 18.95). After extracting the two ECG recordings associated with the lowest and highest s values, we find that both belong to the same class, normal sinus rhythm. Upon closer inspection, the recording with the lower s value exhibits a feature known as ST-elevation. This feature, which is characterized by the elevation of the segment between the S and T waves (deflections) of the ECG signal relative to the baseline, is typically associated with heart attacks. Mapping an ECG recording with such an abnormal feature to the normal class would have been a source of confusion for the network. We provide additional qualitative evidence in Appendix G.

We also leverage s to learn a curriculum (Bengio et al., 2009). First, we fit a Gaussian, $\mathcal{N}(\mu_{\mathcal{T}}, \sigma^2_{\mathcal{T}})$, to each of the distributions in Fig. 6. Using this information, we define the difficulty of task $\mathcal{T}$ as $d_{\mathcal{T}} = \frac{1}{\mu_{\mathcal{T}}}$ and the similarity, $S(j,k)$, between task j and k as shown in Eq. 10. In Fig. 7, we illustrate the resulting pairwise task similarity matrix.

$$S(j,k) = 1 - \underbrace{\sqrt{1 - \sqrt{\frac{2\sigma_0\sigma_1}{\sigma_0^2\sigma_1^2}} e^{-\frac{1}{4}\frac{(\mu_0-\mu_1)^2}{\sigma_0^2\sigma_1^2}}}}_{\mathcal{D}_H = \text{Hellinger Distance}} \quad (10)$$

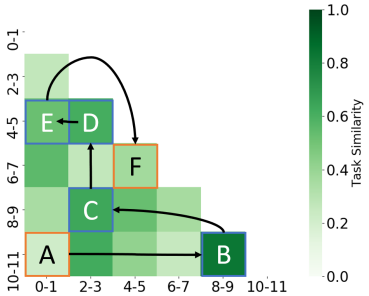


Figure 7: Similarity of tasks in the Class-IL scenario. We create a curriculum by following chaining tasks that are similar to one another.

We design a curriculum by first selecting the easiest task ($\downarrow d_{\mathcal{T}}$) and then creating a chain of tasks that are similar to one another as shown in Fig. 7. For an anti-curriculum, we start with the hardest task ($\uparrow d_{\mathcal{T}}$). In Table 3, we illustrate the performance of various curricula and find that a curriculum exhibits higher constructive interference than a random one ($\text{BWT} = 0.087$ vs. 0.053). Such an outcome aligns well with the expectations of curriculum learning, thus helping to further validate the intuition underlying β.

Table 3: Performance of CLOPS in the Class-IL scenario with different curricula. Storage and acquisition fractions are $b = 0.25$ and $a = 0.50$, respectively. Results are shown across five seeds.

Task Order	Average AUC	BWT	BWT_t	BWT_λ
Random	**0.796 ± 0.013**	0.053 ± 0.023	0.018 ± 0.010	0.008 ± 0.016
Curriculum	0.744 ± 0.009	**0.087 ± 0.011**	**0.038 ± 0.021**	**0.076 ± 0.037**
Anti-curriculum	0.783 ± 0.022	0.058 ± 0.016	(0.013) ± 0.013	(0.003) ± 0.014

7 Discussion and Future Work

In this paper, we introduce a replay-based method applied to physiological signals, entitled CLOPS, to mitigate destructive interference during continual learning. CLOPS consists of an importance-guided buffer-storage and active-learning inspired buffer-acquisition mechanism. We show that CLOPS outperforms the state-of-the-art methods, GEM and MIR, on both backward and forward weight transfer. Furthermore, we propose learnable parameters, as a proxy for the difficulty with which instances are classified, which can assist with quantifying task difficulty and improving network interpretability. We now elucidate future avenues worth exploring.

Extensions to Task Similarity. The notion of task similarity was explored by Thrun & O'Sullivan (1996); Silver & Mercer (1996). In this work, we proposed a definition of task similarity and used it to order the presentation of tasks. The exploration of more robust definitions, their validation through domain knowledge, and their exploitation for generalization is an exciting extension.

Predicting Destructive Interference. Destructive interference is often dealt with in a reactive manner. By *predicting* the degree of forgetting that a network may experience once trained sequentially can help alleviate this problem in a proactive manner.

REFERENCES

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pp. 11849–11860, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019b.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 41–48, 2009.

Richard A Caruana. Multitask connectionist learning. In *In Proceedings of the 1993 Connectionist Models Summer School*. Citeseer, 1993.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Yukun Chen, Robert J Carroll, Eugenia R McPeek Hinz, Anushi Shah, Anne E Eyler, Joshua C Denny, and Hua Xu. Applying active learning to high-throughput phenotyping algorithms for electronic health records data. *Journal of the American Medical Informatics Association*, 20(e2): e253–e259, 2013.

Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 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.

Wenbo Gong, Sebastian Tschiatschek, Richard Turner, Sebastian Nowozin, and José Miguel Hernández-Lobato. Icebreaker: element-wise active information acquisition with bayesian deep latent gaussian model. *arXiv preprint arXiv:1908.04537*, 2019.

Awni Y Hannun, Pranav Rajpurkar, Masoumeh Haghpanahi, Geoffrey H Tison, Codie Bourn, Mintu P Turakhia, and Andrew Y Ng. Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, 25(1):65, 2019.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Thirty-second AAAI Conference on Artificial Intelligence*, 2018.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-III, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016.

Dani Kiyasseh, Tingting Zhu, and David A Clifton. Alps: Active learning via perturbations. *arXiv preprint arXiv:2004.09557*, 2020.

Matthias Lenga, Heinrich Schulz, and Axel Saalbach. Continual learning for domain adaptation in chest x-ray classification. *arXiv preprint arXiv:2001.05922*, 2020.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

E. A. Perez Alday, A. Gu, A. Shah, C. Liu, A. Sharma, S. Seyedi, A. Bahrami Rad, M. Reyna, and G. Clifford. Classification of 12-lead ECGs: the PhysioNet - computing in cardiology challenge 2020 (version 1.0.1). *PhysioNet*, 2020.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, pp. 348–358, 2019.

Shreyas Saxena, Oncel Tuzel, and Dennis DeCoste. Data parameters: A new family of parameters for learning a differentiable curriculum. In *Advances in Neural Information Processing Systems*, pp. 11093–11103, 2019.

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

Daniel L Silver and Robert E Mercer. The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. In *Learning to learn*, pp. 213–233. Springer, 1996.

Asim Smailagic, Pedro Costa, Hae Young Noh, Devesh Walawalkar, Kartik Khandelwal, Adrian Galdran, Mostafa Mirshekari, Jonathon Fagert, Susu Xu, Pei Zhang, 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.

Asim Smailagic, Pedro Costa, Alex Gaudio, Kartik Khandelwal, Mostafa Mirshekari, Jonathon Fagert, Devesh Walawalkar, Susu Xu, Adrian Galdran, Pei Zhang, et al. O-medal: Online active deep learning for medical image analysis. *arXiv preprint arXiv:1908.10508*, 2019.

David Spiegelhalter. Should we trust algorithms? *Harvard Data Science Review*, 2(1), 1 2020. doi: 10.1162/99608f92.cb91a35a. URL `https://hdsr.mitpress.mit.edu/pub/56lnenzj`. https://hdsr.mitpress.mit.edu/pub/56lnenzj.

Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *ICML*, volume 96, pp. 489–497, 1996.

Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Guijin Wang, Chenshuang Zhang, Yongpan Liu, Huazhong Yang, Dapeng Fu, Haiqing Wang, and Ping Zhang. A global and updatable ecg beat classification system based on recurrent neural networks and active learning. *Information Sciences*, 501:523–542, 2019.

Jianwei Zheng, Jianming Zhang, Sidy Danioko, Hai Yao, Hangyuan Guo, and Cyril Rakovski. A 12-lead electrocardiogram database for arrhythmia research covering more than 10,000 patients. *Scientific Data*, 7(1):1–8, 2020.

Zongwei Zhou, Jae Shin, Lei Zhang, Suryakanth Gurudu, Michael Gotway, and Jianming Liang. Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7340–7351, 2017.

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

A CLOPS ALGORITHM

In this section, we outline the various components of CLOPS algorithmically. CLOPS consists of predominantly three main functions: 1) StoreInBuffer, 2) MonteCarloSamples, and 3) AcquireFromBuffer. These are shown in Algorithms 2-4, respectively. StoreInBuffer leverages the task-instance parameters, $\beta_{i\mathcal{T}}$, and the storage function, $s_{i\mathcal{T}}$ (Eq. 6), to store instances into the buffer, $\mathcal{D}_B$, for future replay. MonteCarloSamples performs several Monte Carlo forward passes through the network to allow for the calculation of an acquisition function, α. AcquireFromBuffer implements a user-defined acquisition function, which could be uncertainty-based, in order to determine the relative informativeness of each instances in the buffer. This is then exploited to determine which instances to acquire for replay when the network is training on future tasks.

Algorithm 1: CLOPS

Input: MC epochs τ_{MC}, sample epochs τ_S, MC samples T, storage fraction b, acquisition fraction a, task data $\mathcal{D}_T$, buffer $\mathcal{D}_B$, training epochs per task τ
for $(x, y) \sim \mathcal{D}_T$ **do**
  calculate $\mathcal{L}$ using eq. 5
  update α_{ij}
  if epoch = τ **then**
    $\mathcal{D}_B$ = StoreInBuffer($\beta_{i\mathcal{T}}$, b)
  end if
  if epoch in τ_{MC} **then**
    G = MonteCarloSamples($\mathcal{D}_B$)
  end if
  if epoch in τ_S **then**
    $\mathcal{D}_T$ = AcquireFromBuffer($\mathcal{D}_B$, G, a)
  end if
end for

Algorithm 2: StoreInBuffer

Input: task-instance parameters $\beta_{i\mathcal{T}}$, b
calculate $s_{i\mathcal{T}}$ using eq. 6
SortDescending($s_{i\mathcal{T}}$)
$(x_b, y_b) \subset \mathcal{D}_T$
$\mathcal{D}_B \in (\mathcal{D}_B \cup (x_b, y_b))$

Algorithm 3: MonteCarloSamples

Input: $\mathcal{D}_B$
for $(x, y) \sim \mathcal{D}_B$ **do**
  for MC sample in T **do**
    obtain $p(y|x, \hat{\omega})$ and store in $G \in \mathbb{R}^{M \text{x} T \text{x} C}$
  end for
end for

Algorithm 4: AcquireFromBuffer

Input: $\mathcal{D}_B$, MC posterior distributions G, a
calculate β using eq. 7
SortDescending(β)
$(x_a, y_a) \subset \mathcal{D}_B$
$\mathcal{D}_T \in (\mathcal{D}_T \cup (x_a, y_a))$

B DATASETS

B.1 DATA PREPROCESSING

In this section, we describe in detail each of the datasets used in our experiments in addition to any pre-processing that is applied to them.

Cardiology ECG, $\mathcal{D}_1$ (Hannun et al., 2019). Each ECG recording was originally 30 seconds with a sampling rate of 200Hz. Each ECG frame in our setup consisted of 256 samples resampled to 2500 samples. Labels made by a group of physicians were used to assign classes to each ECG frame depending on whether that label coincided in time with the ECG frame. These labels are: AFIB, AVB, BIGEMINY, EAR, IVR, JUNCTIONAL, NOISE, NSR, SVT, TRIGEMINY, VT, 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.

Chapman ECG, $\mathcal{D}_2$ (Zheng et al., 2020). Each ECG recording was originally 10 seconds with a sampling rate of 500Hz. We downsample the recording to 250Hz and therefore each ECG frame in our setup consisted of 2500 samples. We follow the labelling setup suggested by Zheng et al. (2020) which resulted in four classes: Atrial Fibrillation, GSVT, Sudden Bradychardia, Sinus Rhythm. The ECG frames were not normalized.

PhysioNet 2020 ECG, $\mathcal{D}_3$ (Perez Alday et al., 2020). Each ECG recording varied in duration from 6 seconds to 60 seconds with a sampling rate of 500Hz. Each ECG frame in our setup consisted of 2500 samples (5 seconds). We assign multiple labels to each ECG recording as provided by the original authors. These labels are: AF, I-AVB, LBBB, Normal, PAC, PVC, RBBB, STD, and STE. The ECG frames were normalized in amplitude between the values of 0 and 1.

B.2 DATASET SPLIS

In this section, we outline in Table 7 the number of instances present in each of the training, validation, and test sets of the various datasets. Data were split into these sets according to a 60, 20, 20 configuration and were always performed at the patient-level. In other words, a patient did not appear in more than one set.

Table 4: Number of instances (number of patients) in the training, validation, and test splits for datasets $\mathcal{D}_1$ to $\mathcal{D}_3$. The number of instances for $\mathcal{D}_3$ are shown for one lead.

Dataset	Train	Validation	Test
$\mathcal{D}_1$	4079 (180)	1131 (50)	1386 (62)
$\mathcal{D}_2$	76,606 (6387)	25,535 (2129)	25,555 (2130)
$\mathcal{D}_3$	11,598 (4402)	3238 (1100)	4041 (1375)

B.3 TASK SPLITS

In the main manuscript, we conduct experiments in three continual learning scenarios: 1) Class-IL, 2) Time-IL, and 3) Domain-IL. In this section, we outline the number of instances present in each of the tasks that define the aforementioned scenarios. Data were split across training, validation, and test sets according to patient ID using a 60, 20, 20 configurations. In other words, a patient did not appear in more than one set.

Table 5: Number of instances in the training, validation, and test sets of tasks in our three continual learning scenarios, Class-IL, Time-IL, and Domain-IL.

	Class-IL					
Task Name	0-1	2-3	4-5	6-7	8-9	10-11
Train	781	227	463	2118	309	179
Validation	126	141	118	587	83	82
Test	285	77	130	703	89	102

	Time-IL		
Task Name	Term 1	Term 2	Term 3
Train	37596	12534	12552
Validation	20586	6858	6864
Test	18424	6143	6139

	Domain-IL				
Task Name	Lead I	Lead II	Lead III	...	Lead V6
Train	11598	11598	11598	...	11598
Validation	3238	3238	3238	...	3238
Test	4041	4041	4041	...	4041

C Implementation Details

In this section, we outline the architecture of the neural network used for all experiments. We chose this architecture due to its simplicity and its simultaneous ability to learn on the datasets provided. We also outline the batchsize and the learning rate used for training on the various datasets.

C.1 Network Architecture

Table 6: Network architecture used for all 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 all convolutional layers.

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)

C.2 Experiment Details

Table 7: Batchsize and learning rates used for training with different datasets. The Adam optimizer was used for all experiments.

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

C.3 BASELINE REPLAY-BASED METHODS

In this section, we outline how we have adapted two replay-based methods for application in our continual learning scenarios. This adaptation was necessary since both of these methods were designed for a more extreme online setting whereby instances are only seen once and never again.

GEM. At the end training on each task, we randomly select a subset of the instances and store them in the buffer. Since our data are shuffled, this is equivalent to the strategy proposed by the original authors which involves storing the most recent instances. On subsequent tasks, and at each iteration, we check the gradient dot product requirement outlined by the original authors, and if that is violated, we solve a quadratic programming problem. This implementation is exactly the same as that found in the original MIR manuscript.

MIR. At the end of training on each, we randomly select a subset of the instances and store them in the buffer. This is more suitable for our scenarios relative to reservoir sampling, which was implemented by the original authors. As our approach is task-aware, our buffer is task-aware, unlike the original MIR implementation. Such information would only strengthen the performance of our adapted version. On subsequent tasks, and at each iteration, we randomly select a subset of the instances from the buffer, and score them according to the metric proposed by the original authors. We sort these instances in descending order of the scores and acquire the top scoring instances from each task. We ensure that the total number of instances replayed is equal to the number of instances in the mini-batch of the current task.

D EVALUATION METRICS

Metrics used to evaluate continual learning methodologies are of utmost importance as they provide us with a glimpse of the strengths and weaknesses of various learning strategies. In this section, we outline the traditional metrics used within continual learning. We argue that such metrics are limited in that they only capture the global behaviour of strategies. Consequently, we propose two more fine-grained metrics in the main manuscript.

Average AUC. Let R_j^i represent the performance of the network in terms of AUC on task *j* after having been trained on task *i*.

$$\text{Average AUC} = \frac{1}{N}\sum_{j=1}^{N}\mathrm{R}_j^N \tag{11}$$

Backward Weight Transfer. $\mathrm{BWT} \in [-1, 1]$ is used to quantify the degree to which training on subsequent tasks affects the performance on previously-seen tasks. Whereas positive BWT is indicative of constructive interference, negative BWT is indicative of destructive interference.

$$\mathrm{BWT} = \frac{1}{N-1}\sum_{j=1}^{N-1}\mathrm{R}_j^N - \mathrm{R}_j^j \tag{12}$$

E Comparison of CLOPS to Baseline Methods

In the main manuscript, we conduct experiments in three continual learning scenarios: 1) Class-IL, 2) Time-IL, and 3) Domain-IL. For each, we illustrate the learning curves during training and the corresponding evaluation metric values. In this section, we present results that are complementary to those found in the main manuscript. More specifically, we compare the performance of CLOPS to that of the baseline methods implemented.

E.1 Time-IL

In the Time-IL scenario, CLOPS performs on par with the fine-tuning strategy. This can be seen in Fig. 8 across all evaluation metrics. As noted in the main manuscript, the utility of CLOPS in this scenario lies in forward weight transfer, behaviour that is better observed in the learning curves in Fig. 4 of Sec. 6.2. Despite the dominance of multi-task learning with an $\mathrm{AUC} = 0.971$, its implementation is the least feasible particularly in the Time-IL setting where data are collected at different time intervals during the year.

Table 8: Performance of CL strategies in the Time-IL scenario. Storage and acquisition fractions are $b = 0.25$ and $a = 0.50$, respectively. Mean and standard deviation values are shown across five seeds.

Method	Average AUC	BWT	BWT_t	BWT_λ
MTL	**0.971 ± 0.006**	-	-	-
Fine-tuning	0.824 ± 0.004	(0.020) ± 0.005	(0.007) ± 0.003	0.010 ± 0.001
Replay-based Methods				
GEM	*Could not be solved*			
MIR	0.856 ± 0.010	(0.007) ± 0.006	(0.003) ± 0.004	0.001 ± 0.004
CLOPS	0.834 ± 0.014	(0.018) ± 0.004	(0.007) ± 0.003	0.007 ± 0.003

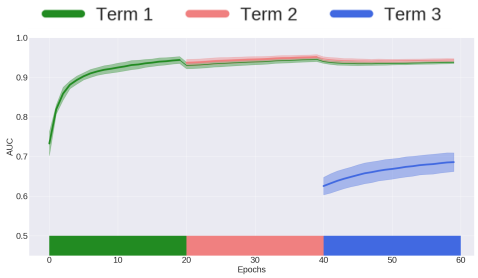
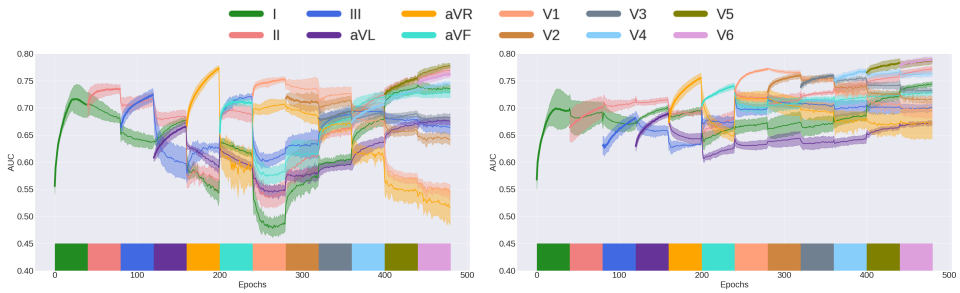


Figure 8: Mean validation AUC of MIR in the Time-IL scenario. Results are shown as a function of storage fractions, *b*, and acquisition fractions, *a* and are an average across five seeds.

E.2 DOMAIN-IL

To gain a better understanding of the learning dynamics of CLOPS in the domain-IL scenario, we plot the validation AUC in Fig. 9. Here, significant destructive interference occurs in the fine-tuning strategy. This is shown by large drops in the AUC of one task when subsequent tasks are trained on. For instance, when training on Lead V1 (starting at epoch 240), performance on Lead I (green) drops from an $\mathrm{AUC} = 0.725 \rightarrow 0.475$, completely erasing any progress that had been made on that lead. We hypothesize that this is due to the different representations of instances belonging to Lead V1 and Lead I. Anatomically speaking, these two leads are projections of the same electrical signal in the heart that are oriented approximately 180 degrees to one another. This behaviour is absent for almost all leads when CLOPS is implemented, as shown in Fig. 9b. A notable exception is Lead aVR at epoch 200 where performance drops from an $\mathrm{AUC} \approx 0.75 \rightarrow 0.65$.



(a) Fine-tuning

(b) CLOPS

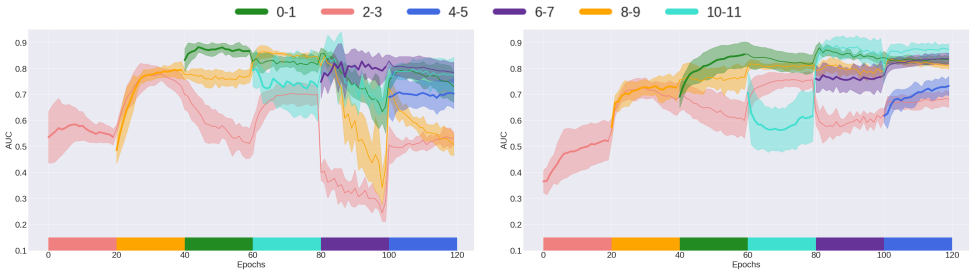
Figure 9: Mean validation AUC of a) fine-tuning and b) CLOPS ($b = 0.25$ and $a = 0.50$) strategy in the Domain-IL scenario. Each task belongs to the same dataset yet different input modality. Coloured blocks indicate tasks on which the learner is currently being trained. The shaded area represents one standard deviation from the mean across 5 seeds.

F Effect of Task Order

The order in which tasks are presented to a continual learner can significantly impact the degree of destructive interference. To quantify this phenomenon and evaluate the robustness of CLOPS to task order, we repeat the Class-IL experiment conducted in the main manuscript after having randomly shuffled the tasks.

F.1 Class IL

Task order does have an effect on destructive interference. By comparing the evaluation metric values shown in Table 9 and Table 1 of Sec. 6.1, we show that poorer generalization performance is achieved when learning sequentially on tasks that have been randomly shuffled. Given that one has limited control over the order in which data is streamed, continual learning approaches must be robust to task order. Indeed, we claim that CLOPS is robust to such changes. This can be seen by its achievement of an $\text{AUC} = 0.796$ regardless of task order (in Tables 1 and 9).



(a) Fine-tuning (b) CLOPS

Figure 10: Mean validation AUC of a) fine-tuning strategy and b) CLOPS ($b = 0.25$ and $a = 0.50$) in the Class-IL scenario with 6 tasks after being *randomly re-ordered*. Each task belongs to a mutually-exclusive pair of classes from $\mathcal{D}_1$. Coloured blocks indicate tasks on which the learner is currently being trained. The shaded area represents one standard deviation from the mean across 5 seeds.

Table 9: Performance of CL strategies in the Class-IL scenario. Storage and acquisition fractions are $b = 0.25$ and $a = 0.50$, respectively. Results are shown across five seeds. Values enclosed in parentheses are negative.

Method	Average AUC	BWT	BWT_t	BWT_λ
Fine-tuning	0.672 ± 0.022	$(0.081) \pm 0.049$	0.014 ± 0.034	$(0.055) \pm 0.025$
CLOPS	$\mathbf{0.796 \pm 0.006}$	$\mathbf{0.110 \pm 0.021}$	$\mathbf{0.096 \pm 0.025}$	$\mathbf{0.095 \pm 0.036}$

Distribution of Task-Instance Parameters. We also illustrate the distribution of the tracked task-instance parameters in Fig. 11. These distributions differ in terms of overlap compared to those in Fig. 6 of Sec. 6.1. They both, however, follow a Gaussian distribution and roughly maintain their relative locations to one another. For instance, task $[6, 7]$ and $[10, 11]$ consistently generate the lowest and highest s values, respectively, regardless of task order. Such a finding illustrates that task-instance parameters are agnostic to task-order, which is a desirable trait when attempting to quantify task difficulty.

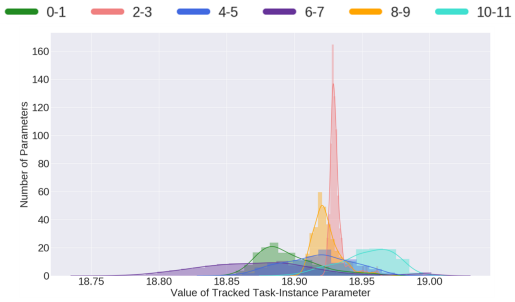


Figure 11: Distribution of the tracked task-instance parameter values, s, corresponding to our proposed strategy in the Class-IL scenario with 6 tasks. Storage and acquisition fractions are $b = 0.25$ and $a = 0.50$, respectively. Each colour corresponds to a different task. Results are shown for one seed.

G QUALITATIVE EVALUATION OF TASK-INSTANCE PARAMETERS, β

In this section, we aim to qualitatively evaluate the interpretation of task-instance parameters as proxies for task-difficulty. To do so for the various CL scenarios, we plot two ECG tracings that correspond to the highest and lowest s values (see eq. 6). As instances with low s values should be more difficult to classify, these ECG tracings might be expected to exhibit abnormalities that make it difficult for a cardiologist to correctly diagnose. Conversely, ECG tracings with high s values should be easy to classify from an expert's perspective. We illustrate these tracings, which are coloured based on the task to which they belong, for the Time-IL and Domain-IL scenarios.

G.1 TIME-IL

In Fig. 12, we see that the network had a difficult time classifying the ECG tracing that corresponds to the lowest s value with a ground truth label of GSVT (Supra-ventricular Tachycardia). On the other end of the spectrum, the network was able to comfortably classify the ECG tracing that corresponds to the highest s value with a ground truth label of SB (Sudden Bradycardia).

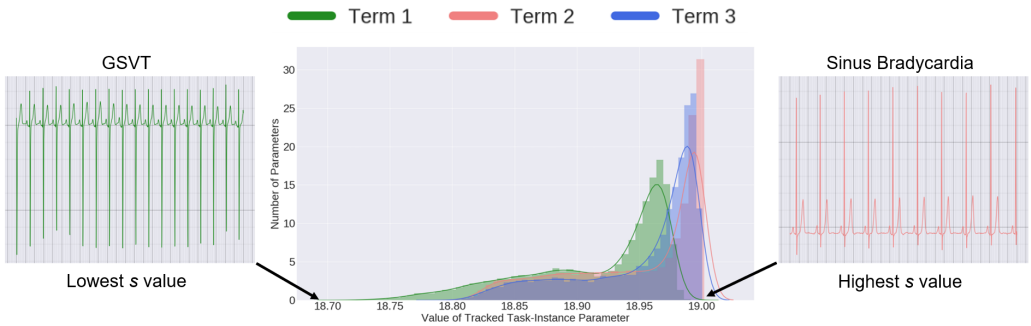


Figure 12: Distribution of the tracked task-instance parameter values, s, corresponding to our proposed strategy in the Time-IL scenario. Each colour corresponds to a different task. Results are shown for one seed. The ECG tracings correspond to the lowest and highest s values.

G.2 DOMAIN-IL

In Fig. 13, we see that the network had a difficult time classifying the ECG tracing that corresponds to the lowest s value with a ground truth label of AF (Atrial Fibrillation). This could be due to the amount of noise present in the tracing. On the other end of the spectrum, the network was able to comfortably classify the ECG tracing that corresponds to the highest s value with a ground truth label of AF also.

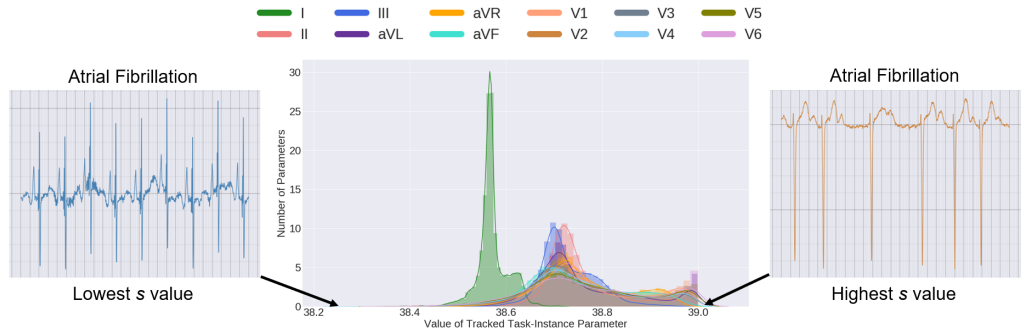


Figure 13: Distribution of the tracked task-instance parameter values corresponding to our proposed strategy in the Domain-IL scenario. Each colour corresponds to a different task. Results are shown for one seed. The ECG tracings correspond to the lowest and highest s values.

H Effect of Storage Function Form

In the main manuscript, we proposed the following storage function as a way to guide the storage of instances into the replay buffer. This function is dependent upon the task-instance parameters, β, as they have been tracked throughout the training process.

$$s_{i\mathcal{T}} = \int_0^{\tau} \beta_{i\mathcal{T}}(t)dt \approx \sum_{t=0}^{\tau} \left(\frac{\beta_{i\mathcal{T}}(t+\Delta t) + \beta_{i\mathcal{T}}(t)}{2} \right) \Delta t \tag{13}$$

In this section, we aim to quantify the effect of the form of the storage function on the generalization performance of our network. More specifically, we explore a different form of the storage function where the task-instance parameters, β, are squared before the trapezoidal rule is applied. Mathematically, the storage function now takes on the following form.

$$s_{i\mathcal{T}} = \int_0^{\tau} \beta^2_{i\mathcal{T}}(t)dt \approx \sum_{t=0}^{\tau} \left(\frac{\beta^2_{i\mathcal{T}}(t+\Delta t) + \beta^2_{i\mathcal{T}}(t)}{2} \right) \Delta t \tag{14}$$

To compare these two storage functions, we conduct experiments in the Class-IL scenario as we vary the storage fraction, b, and acquisition fraction, a. In Fig. 14, we present the validation AUC of these two experiments conducted across five seeds. We find that our proposed storage function, and the one used throughout the manuscript (Eq. 13), is more advantageous than that shown in Eq. 14. This is evident by the consistently higher Average AUC scores. A possible explanation for this observation is the following. Recall that task-instance parameters, β, are a rough proxy for the difficulty of an instance to be classified. Therefore, when ranking such instances based on Eq. 13 for storage purposes, we are effectively storing the relatively 'easiest-to-classify' instances into the buffer. We hypothesize that upon squaring the task-instance parameters, as is done in Eq. 14, this interpretation no longer holds. As a result, the discrepancy between instances diminishes and thus hinders the storage process.

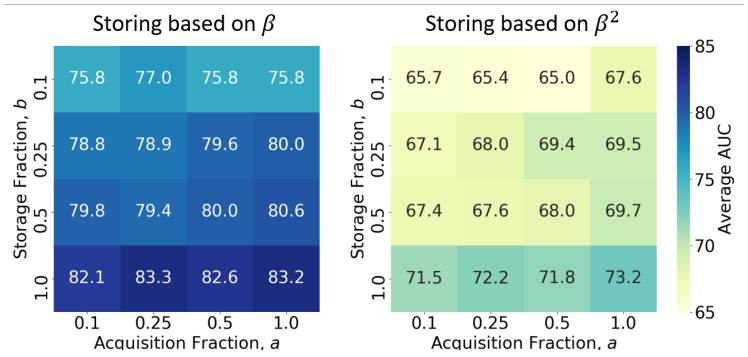


Figure 14: Effect of form of storage function on the generalization performance of the network. Mean validation AUC when buffer storage is based on **(Left)** Eq. 13 and **(Right)** Eq. 14. We show that our proposed storage function, and the one used throughout the manuscript (Eq. 13), is more advantageous than that shown in Eq. 14.

I Effect of Storage Fraction, b, and Acquisition Fraction, a, on Performance

Replay-based continual learning strategies can be computationally expensive and resource-hungry due to the presence of a buffer and the need to acquire instances from it. To map out the performance of CLOPS under various resource constraints, we set out to investigate the effect of changes in the storage and acquisition fractions on the various evaluation metrics.

To simultaneously validate our decision of storing the top b instances from each task into the buffer, we conduct the aforementioned experiments in two scenarios: 1) The first scenario involves sorting the instances according to their s value (eq. 6) and storing the top b fraction of instances into the buffer **(Storing top *b* fraction)**. 2) The second scenario involves storing the bottom b fraction of instances **(Storing bottom *b* fraction)**. By conducting this specific experiment, we look to determine the relative benefit of replaying either relatively easy or difficult instances.

I.1 Average AUC

Larger storage and acquisition fractions should further alleviate destructive interference and improve generalization performance. The intuition is that large fractions will expose the learner to a more representative distribution of data from previous tasks. We quantify this graded response in Fig. 15 where the $\text{AUC} = 0.758 \rightarrow 0.832$ as $b, a = 0.1 \rightarrow 1$. We also claim that a performance bottleneck lies at the storage phase. This can be seen by the larger improvement in performance as a result of an increased storage fraction compared to that observed for a similar increase in acquisition fraction. Despite this, a strategy with fractions as low as $b = 0.25$ and $a = 0.1$ is sufficient to outperform the fine-tuning strategy.

In addition to exploring the graded effect of fraction values, we wanted to explore the effect of storing the bottom, most difficult, b fraction of instances in a buffer. The intuition is that if a learner can perform well on these difficult replayed instances, then strong performance should be expected on the remaining relatively easier instances. We show in Fig. 15 (right) that although the performance seems to be on par with that in Fig. 15 (left) for $b = (0.5, 1)$, the most notable differences arise at fractions $b < 0.5, a < 0.5$ (red box). We believe this is due to the extreme 'hard-to-classify' nature of instances with low s values. These findings justify our storing of the top b fraction of instances.

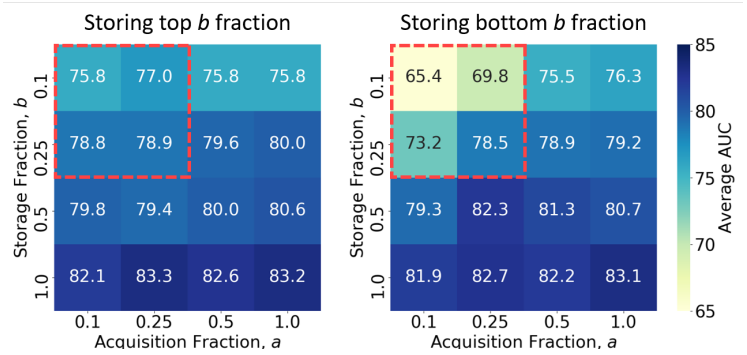
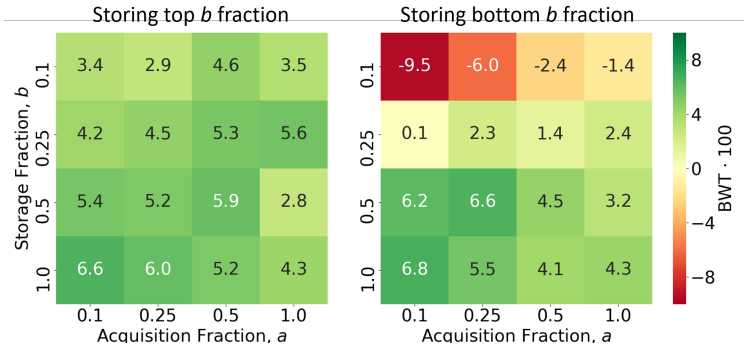


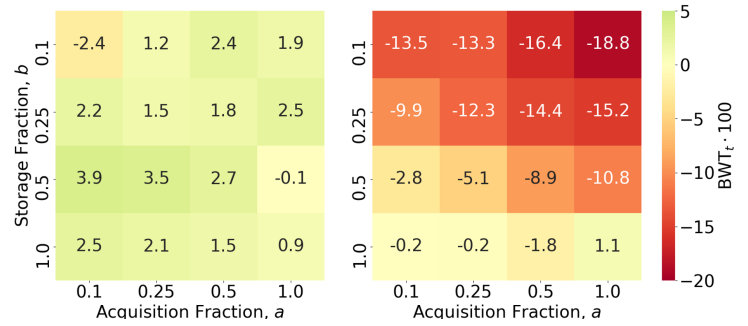
Figure 15: Mean validation AUC of CLOPS in the Class-IL scenario implemented while either storing b instances with the highest s value (left) or lowest s value (right). Please refer to eq. 6 for the definition of s. Results are shown as a function of storage fractions, b, and acquisition fractions, a and are an average across five seeds. Darker coloured cells indicate higher generalization performance.

I.2 Backward Weight Transfer

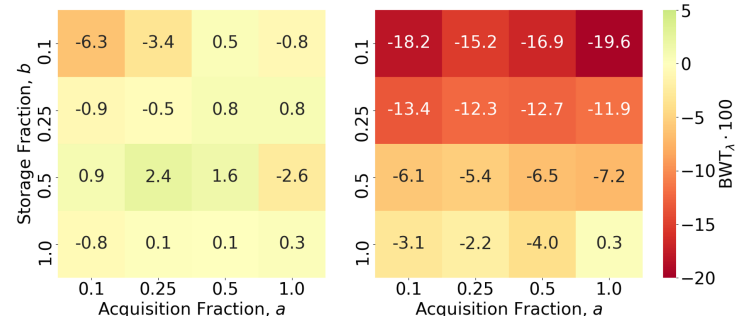
In this section, we illustrate the results of the two scenarios 1) Storing top b fraction and 2) Storing bottom b fraction on backward weight transfer. In the top left heatmap, we show that as the storage fraction $b = 0.1 \rightarrow 1$ at an acquisition fraction, $a = 0.1$, is associated with improved constructive interference (BWT $= 0.034 \rightarrow 0.066$). Although this is also true for the scenario in the second column, we show that performance is worse in this case. Such a finding corroborates our claim in the main manuscript that storing the top instances, which correspond to the 'easiest-to-classify' instances, is more beneficial in the context of CL. This provides further evidence that supports our use of the buffer-storage strategy represented by the first column for all experiments conducted in the main manuscript.



(a) BWT



(b) BWT$_t$



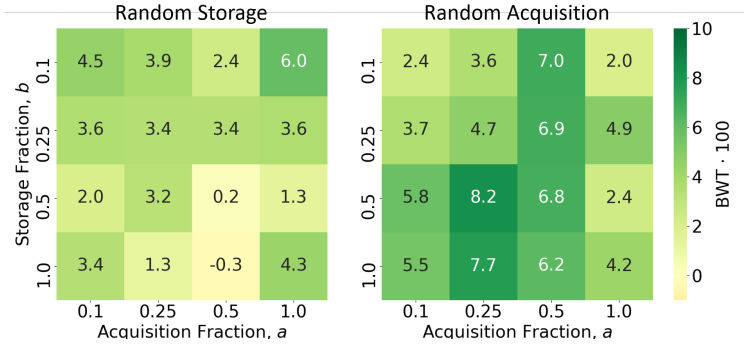
(c) BWT$_\lambda$

Figure 16: Backward weight transfer on the validation set using CLOPS while storing top b fraction of instances (left column) and bottom b fraction of instances (right column) in the Class-IL scenario. Results are shown as a function of storage fractions, b, and acquisition fractions, a and are an average across five seeds.

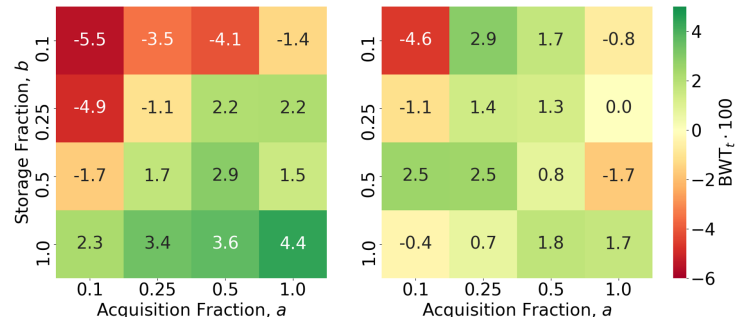
J Effect of Task-Instance Parameters, β, and Acquisition Function, α

In this section, we illustrate the effect of two ablation studies on backward weight transfer. **Random Storage** (RS) is a training procedure that stores instances randomly into a buffer yet acquires them using an acquisition function. **Random Acquisition** (RA), on the other hand, employs task-instance parameters for buffer-storage yet acquires instances randomly from the buffer.

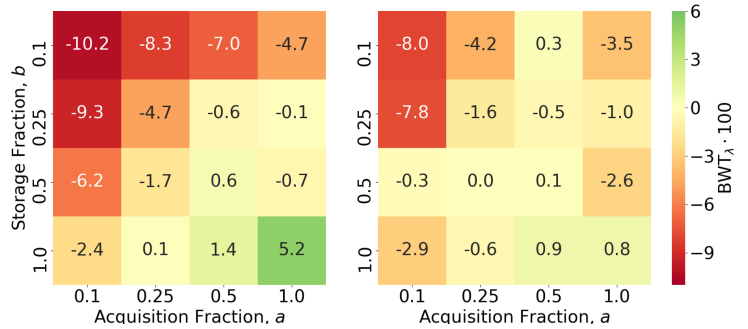
When comparing the BWT_λ heatmaps, we show that the random storage scenario leads to greater destructive interference compared to random acquisition. This can be seen at low storage and acquisition fractions ($b < 0.5$ and $a < 0.5$). Since RA and RS can be thought of as 'smart' storage and 'smart' acquisition, respectively, the superiority of the former implies that a storage strategy is more important than an acquisition strategy in this context and when evaluated from the perspective of backward weight transfer. With most recent CL replay strategies solely focusing on acquisition, our finding suggests that further research into storage functions could be of value.



(a) BWT



(b) BWT_t



(c) BWT_λ

Figure 17: Backward weight transfer on the validation set using a Random Storage (left column) and Random Acquisition (right column) strategy in the Class-IL scenario. Results are shown as a function of storage fractions, b, and acquisition fractions, a and are an average across five seeds.

K Effect of Weighting Replayed Instances

In the main manuscript, we stated that replayed instances are *not* weighted with task-instance parameters. Our hypothesis was that this would negatively interfere with the learning process on subsequent tasks. To quantify the effect of such a weighting, we perform several experiments in which replayed instances are weighted according to their corresponding task-instance parameters. Notably, we freeze these parameters and no longer update them on subsequent tasks. In other words, their previous dual role as a weighting and buffer-storage mechanism now collapses to the former only. In Table 10, we illustrate the performance of CLOPS with and without the weighting of replayed instances using the task-instance parameters.

We find that weighting *replayed* instances with frozen task-instance parameters is a detriment to backward weight transfer. For example, in the Class-IL scenario, $\mathrm{BWT} = 0.053$ and 0.042 for CLOPS without and with the weighting coefficient, respectively. This can be seen across the continual learning scenarios. We hypothesize that this behaviour is due to the 'down-weighting' of replayed instances. More specifically, since task-instance parameters, $\beta < 1$, networks end up learning less from replayed instances.

Table 10: Performance of CL strategies in the three continual learning scenarios with and without weighted replayed instances. Storage and acquisition fractions are $b = 0.25$ and $a = 0.50$, respectively. Mean and standard deviation are shown across five seeds.

Method	Average AUC	BWT	BWT_t	BWT_λ
Class-IL				
CLOPS	0.796 ± 0.013	**0.053 ± 0.023**	**0.018 ± 0.010**	**0.008 ± 0.016**
CLOPS weighted	**0.800 ± 0.006**	0.042 ± 0.020	0.005 ± 0.014	$(0.014) \pm 0.016$
Time-IL				
CLOPS	**0.834 ± 0.014**	**(0.018) ± 0.004**	**(0.007) ± 0.003**	0.007 ± 0.003
CLOPS weighted	0.818 ± 0.012	$(0.024) \pm 0.012$	$(0.011) \pm 0.006$	0.007 ± 0.002
Domain-IL				
CLOPS	0.731 ± 0.001	**(0.011) ± 0.002**	**(0.020) ± 0.004**	**(0.019) ± 0.009**
CLOPS weighted	**0.741 ± 0.007**	$(0.016) \pm 0.007$	$(0.024) \pm 0.001$	$(0.024) \pm 0.003$

L EFFECT OF NUMBER OF MONTE CARLO SAMPLES, *T*

The functionality of uncertainty-based acquisition functions such as $\mathrm{BALD}_{\mathrm{MCD}}$ is dependent upon a reasonable approximation of the region of uncertainty in the hypothesis space. Improved approximations are usually associated with an increased number of Monte Carlo samples. In all experiments so far, $T = 20$ MC samples were used. In this section, we repeat a subset of these experiments with $T = (5, 10, 50)$ and illustrate their results in Table 11.

As expected, there exists a proportional relationship between the number of MC samples, *T*, and the performance of the network in terms of average AUC and backward weight transfer. For instance, as $T = 5 \rightarrow 50$, the $\mathrm{BWT} = 0.045 \rightarrow 0.061$. This observation implies that an improved approximation of the region of uncertainty can lead to the acquisition of instances from the buffer that increase the magnitude of *constructive* interference.

Table 11: Performance of our strategy in the Class-IL scenario with different numbers of MC samples, *T*. Storage and acquisition fractions are $b = 0.25$ and $a = 0.50$, respectively. Results are shown across five seeds.

MC Samples *T*	Average AUC	BWT	BWT_t	BWT_λ
5	0.765 ± 0.013	0.045 ± 0.019	0.011 ± 0.016	0.002 ± 0.024
10	0.781 ± 0.016	0.051 ± 0.020	0.014 ± 0.018	0.002 ± 0.024
20	$\mathbf{0.796 \pm 0.013}$	0.053 ± 0.023	0.018 ± 0.010	0.008 ± 0.016
50	0.785 ± 0.023	$\mathbf{0.061 \pm 0.009}$	$\mathbf{0.023 \pm 0.008}$	$\mathbf{0.008 \pm 0.012}$

M Effect of Type of Acquisition Function, α

Our modular buffer-acquisition strategy provides researchers with the flexibility to choose their acquisition function of interest. Beyond $\mathrm{BALD}_{\mathrm{MCD}}$, we repeat a subset of our experiments using two recently-introduced acquisition functions, $\mathrm{BALD}_{\mathrm{MCP}}$ and $\mathrm{BALC}_{\mathrm{KLD}}$, which acquire instances based on how sensitive a network is to their perturbed counterpart (Kiyasseh et al., 2020). We define these functions below and illustrate the results in Table 12.

M.1 Definitions of Acquisition Functions

$$\begin{aligned}\textbf{BALD}_{\textbf{MCP}} &= \mathrm{JSD}(p_1, p_2, \ldots, p_T) \\ &= \mathrm{H}(p(y|x)) - \mathbb{E}_{p(z|D_{train})}\left[\mathrm{H}(p(y|x,z))\right]\end{aligned} \tag{15}$$

$$\begin{aligned}\mathrm{H}(p(y|x)) &= \mathrm{H}\left(\int p(y|z)p(z|x)dz\right) \\ &= \mathrm{H}\left(\int p(y|z)q_\phi(z|x)dz\right) \\ &\approx \mathrm{H}\left(\frac{1}{T}\sum_{t=1}^{T} p(y|\hat{z}_t)\right)\end{aligned} \tag{16}$$

where z represents the perturbed input, T is the number of Monte Carlo samples, and $\hat{z}_t \sim q_\phi(z|x)$ is a sample from some perturbation generator.

$$\begin{aligned}\mathbb{E}_{p(z|D_{train})}\left[\mathrm{H}(p(y|x,z))\right] &= \mathbb{E}_{q_\phi(z|x)}\left[\mathrm{H}(p(y|x,z))\right] \\ &\approx \frac{1}{T}\sum_{t=1}^{T}\left[\mathrm{H}(p(y|\hat{z}_t))\right] \\ &= \frac{1}{T}\sum_{t=1}^{T}\left[-\sum_{c=1}^{C} p(y=c|\hat{z}_t)\log p(y=c|\hat{z}_t)\right]\end{aligned} \tag{17}$$

$$\textbf{BALC}_{\textbf{KLD}} = \mathcal{D}_{KL}(\mathcal{N}(\mu(x), \Sigma(x) \parallel \mathcal{N}(\mu(z), \Sigma(z))) \tag{18}$$

where $\mu = \frac{1}{T}\sum_{t=1}^{T} p(y|\hat{\omega}_t, x)$ is the empirical mean of the posterior distributions across T MC samples and $\hat{\omega} \sim q_\theta(\omega)$ represents parameters sampled from the MC distribution as in Gal et al. (2017). $\Sigma = (Y-\mu)^T(Y-\mu)$ is the empirical covariance matrix of the posterior distributions where $Y \in \mathbb{R}^{TxC}$ is the matrix of posterior distributions for all MC samples.

M.2 Results

We show that the type of acquisition function can have a large effect on the final performance of a CL algorithm and the degree of constructive interference that is experienced. Such a finding justifies our use of $\mathrm{BALD}_{\mathrm{MCD}}$ for all experiments conducted in the main manuscript.

Table 12: Performance of CLOPS ($b = 0.25$ and $a = 0.5$) in the Class-IL scenario with different acquisition functions. Mean and standard deviation values are shown across five seeds.

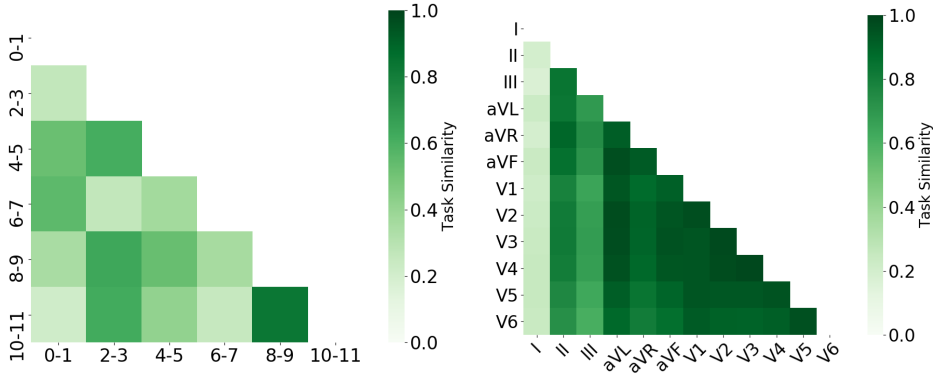
Acquisition Function	Average AUC	BWT	BWT_t	BWT_λ
$\mathrm{BALD}_{\mathrm{MCD}}$	**0.796 ± 0.013**	0.053 ± 0.023	**0.018 ± 0.010**	**0.008 ± 0.016**
$\mathrm{BALD}_{\mathrm{MCP}}$	0.674 ± 0.042	**0.068 ± 0.052**	(0.107 ± 0.035)	(0.091 ± 0.034)
$\mathrm{BALC}_{\mathrm{KLD}}$	0.716 ± 0.039	(0.036 ± 0.035)	(0.104 ± 0.039)	(0.104 ± 0.051)

N QUANTIFYING TASK SIMILARITY VIA TASK-INSTANCE PARAMETERS, β

In this section, we quantify task-similarity and illustrate the similarity matrices between each pair of tasks in our three continual learning scenarios. Given two Gaussians fit to the task-specific distributions of s associated with task j and k (see Fig. 6), parameterized by μ_0, σ_0^2 and μ_1, σ_1^2, respectively, we calculate their pairwise similarity as follows:

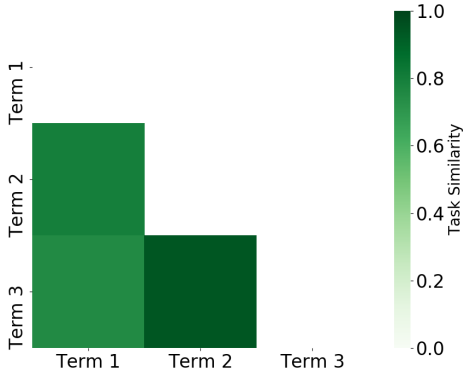
$$S(j,k) = 1 - \underbrace{\sqrt{1 - \sqrt{\frac{2\sigma_0\sigma_1}{\sigma_0^2\sigma_1^2}} e^{-\frac{1}{4}\frac{(\mu_0-\mu_1)^2}{\sigma_0^2\sigma_1^2}}}}_{\mathcal{D}_H = \text{Hellinger Distance}} \tag{19}$$

We apply eq. 10 to all pairs of tasks in each of the continual learning scenarios. This results in the similarity matrices in Fig. 18. In Fig. 7, for instance, we show that task $[8, 9]$ is most similar to task $[10, 11]$. From a clinical perspective and with the appropriate know-how, this information can be used to identify differences between medical conditions, patient cohorts, etc. From a machine learning perspective, such an outcome may prompt further investigation as to why the network views these tasks as being similar. Consequently, we believe these similarity matrices can offer a form of interpretability for both medical and machine learning practitioners.



(a) Class-IL

(b) Domain-IL



(c) Time-IL

Figure 18: Task similarity matrix in the three continual learning scenario: a) Class-IL, b) Domain-IL, and 3) Time-IL using CLOPS ($b = 0.25$ and $a = 0.50$). Pairwise task-similarity is calculated using eq. 10. Results are averaged across five seeds.

O EFFECT OF HYPERPARAMETERS ON MIR

In this section, we aim to provide a more holistic evaluation of our adaptation of MIR. To do so, we vary the two hyper-parameters involved in the implementation and observe their impact on the performance of the models in the Class-IL scenario. These hyper-parameters include the number of instances acquired from the buffer to perform the MIR search (Acquisition Fraction) and the ratio of replayed to current-task instances in each mini-batch (Ratio). In Fig. 14, we present the performance of MIR as a function of the acquisition fraction. In Fig. 13, we present the performance of MIR as a function of the ratio of replayed to current-task instances in each mini-batch.

O.1 ACQUISITION FRACTION

Table 13: Performance of MIR in the Class-IL scenario with different acquisition fractions. Larger acquisition fractions mean that a larger subset of the buffer is searched for maximally-interfered instances. The ratio of replayed to current-task instances is fixed at 1. Mean and standard deviation values are shown across five seeds.

Acquisition Fraction	Average AUC	BWT	BWT_t	BWT_λ
0.1	0.799 ± 0.006	0.046 ± 0.022	$(0.005) \pm 0.031$	$(0.026) \pm 0.028$
0.25	$\mathbf{0.807 \pm 0.010}$	0.043 ± 0.017	0.013 ± 0.016	$(0.007) \pm 0.022$
0.5	0.753 ± 0.014	0.009 ± 0.018	0.001 ± 0.025	$(0.046) \pm 0.022$
1	0.800 ± 0.013	$\mathbf{0.063 \pm 0.023}$	$\mathbf{0.039 \pm 0.024}$	$\mathbf{0.021 \pm 0.023}$

O.2 RATIO OF REPLAYED TO CURRENT-TASK INSTANCES

Table 14: Performance of MIR in the Class-IL scenario with different ratios of replayed instances to current-task instances in the mini-batch. Larger ratios mean that greater emphasis is placed on replayed instances than on current-task instances. The acquisition fraction is fixed at 0.5. Mean and standard deviation values are shown across five seeds.

Ratio	Average AUC	BWT	BWT_t	BWT_λ
1:2	$\mathbf{0.801 \pm 0.013}$	$\mathbf{0.065 \pm 0.025}$	0.011 ± 0.043	$(0.012) \pm 0.030$
1:1	0.753 ± 0.014	0.009 ± 0.018	0.001 ± 0.025	$(0.046) \pm 0.022$
2:1	0.763 ± 0.012	0.016 ± 0.010	0.024 ± 0.017	$(0.020) \pm 0.021$
4:1	0.752 ± 0.016	0.020 ± 0.021	0.030 ± 0.016	$(0.013) \pm 0.021$