Stabilizing Linear Passive-Aggressive Online Learning with Weighted Reservoir Sampling

Skyler Wu
Booz Allen Hamilton
Harvard University
Stanford University
`wu_skyler@bah.com`

Fred Lu
Booz Allen Hamilton
University of Maryland, Baltimore County
`lu_fred@bah.com`

Edward Raff
Booz Allen Hamilton
University of Maryland, Baltimore County
`raff_edward@bah.com`

James Holt
Laboratory for Physical Sciences
`holt@lps.umd.edu`

Abstract

Online learning methods, like the seminal Passive-Aggressive (PA) classifier, are still highly effective for high-dimensional streaming data, out-of-core processing, and other throughput-sensitive applications. Many such algorithms rely on fast adaptation to individual errors as a key to their convergence. While such algorithms enjoy low theoretical regret, in real-world deployment they can be sensitive to individual outliers that cause the algorithm to over-correct. When such outliers occur at the end of the data stream, this can cause the final solution to have unexpectedly low accuracy. We design a weighted reservoir sampling (WRS) approach to obtain a stable ensemble model from the sequence of solutions without requiring additional passes over the data, hold-out sets, or a growing amount of memory. Our key insight is that good solutions tend to be error-free for more iterations than bad solutions, and thus, the number of passive rounds provides an estimate of a solution's relative quality. Our reservoir thus contains K previous intermediate weight vectors with high survival times. We demonstrate our WRS approach on the Passive-Aggressive Classifier (PAC) and First-Order Sparse Online Learning (FSOL), where our method consistently and significantly outperforms the unmodified approach. We show that the risk of the ensemble classifier is bounded with respect to the regret of the underlying online learning method.

1 Introduction

Online learning algorithms are especially attractive when working with high-volume and high-dimensional streaming data, out-of-core processing, and other throughput-sensitive applications [1]. For example, the seminal Vowpal Wabbit uses importance-weighted online learning algorithms [2] to reach high quality solutions quickly, with an optional second pass using LBFGS to refine the solution. The MOA library still uses the Pegasos algorithm as its linear classifier [3]. Most relevantly, online learning algorithms are particularly appealing for binary classification tasks, such as web spam classification [1]. Such algorithms often enjoy fast theoretical convergence rates due to their fast adaptation to errors on individual data points, as opposed to batch or offline learning.

However, in real-world deployment, **online algorithms can be very sensitive to noisy observations in the data stream and over-correct, resulting in out-of-sample performance dropping precipitously between timesteps.** Indeed, in many cases (see Figure 1), an online learning algorithm might

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

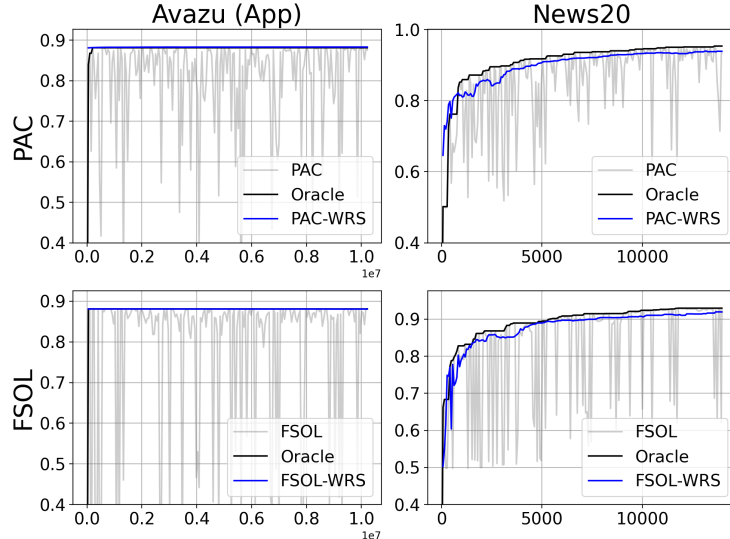



Figure 1: Test accuracies (y-axis) over timestep (x-axis) for PAC-WRS and FSOL-WRS on Avazu (App) and News20. **Light grey lines:** test accuracies of the baseline methods — PAC or FSOL — at each timestep. **Solid black lines:** test accuracies of the "oracle" models, computed as the cumulative maximum of the baselines. **Solid blue lines:** test accuracies of WRS-enhanced models. Note massive fluctuations of grey lines and stability of blue lines. All variants shown are using standard sampling weights for WRS, with simple-averaging.

achieve over 90% test accuracy after a given timestep, but then see its test accuracy drop by $20 - 30\%$ after over-correcting on the next observation in the data stream. In many real-world settings, it may be infeasible computationally or memory-wise to maintain a hold-out evaluation set to select the highest-performance solutions learned by our online algorithm. It may also be practically infeasible to train our classifier over multiple passes of a given dataset or when online algorithms are used for "any-time" ready predictions.

In this paper, we introduce a weighted reservoir sampling (WRS) [4] based approach that dramatically mitigates the aforementioned accuracy fluctuations. Our proposed method, WRS-Augmented Training (WAT), neither requires a hold-out evaluation set nor additional passes over the training data. Most importantly, WAT can be used to stabilize *any* passive-aggressive online learning algorithm. We demonstrate the promise of our WAT method on the Passive Aggressive Classifier (PAC) [5] and First-Order Sparse Online Learning (FSOL) [1] methods — creating two new methods PAC-WRS and FSOL-WRS. **Strikingly, across 16 benchmark datasets, WAT is able to mitigate test accuracy fluctuations in all 16 datasets on FSOL and 14 datasets on PAC.** To analyze the theoretical effectiveness of our method, we situate our approach in the online-to-batch conversion literature, enabling us to obtain generalization bounds on i.i.d. data streams.

2 Review of related work

Our work is motivated in part by a real-world need in malware detection [6], in which large datasets make online methods particularly attractive [7, 8], and a naturally noisy labeling process inhibits standard passive-aggressive methods [9–12]. We show results for the EMBER malware benchmark in Appendix C.

Online learning. In online learning for linear binary classification, one maintains a solution vector $\mathbf{w}_t \in \mathbb{R}^D$, where D is the dimensionality of our data points (usually quite large), and t represents the timestep. At each timestep t, we observe a single observation $(\boldsymbol{x}_t, y_t)$ from a high-throughput data stream, with feature vector $\boldsymbol{x}_t \in \mathbb{R}^D$ and class label $y_t \in \{+1, -1\}$. Given $(\boldsymbol{x}_t, y_t)$, an online learning algorithm will make a minor (potentially no effect) update to $\mathbf{w}_t$ to output $\mathbf{w}_{t+1}$, before receiving the next $(\boldsymbol{x}_{t+1}, y_{t+1})$ in the data stream. The classification rule using $\mathbf{w}_t$ on any test point $\boldsymbol{x}^*$ is simply $\hat{y}^* = \text{sign}(\mathbf{w}_t^\top \boldsymbol{x}^*)$. The goal is that as $t \to \infty$, the sequence of $\mathbf{w}_t$ will enjoy

2

low cumulative loss. Towards this end, many online learning algorithms with various update rules have been proposed in the literature, including the Passive-Aggressive Classifier (PAC) [5], the Adaptive Regularization of Weight Vectors (AROW) methods [13] and the Adaptive Subgradient method (ADAGRAD) [14]. Some online learning algorithms have also been designed to specifically learn sparse solutions for $\mathbf{w}_t$ (proportion of zero entries), such as Truncated Gradient method [15], Stochastic MIrror Descent Algorithm made Sparse method (SMIDAS) [16], Regularized Dual Averaging (RDA) method [17], and First-Order and Second-Order Sparse Online Learning methods (FSOL and SSOL) [1]. In general, these sparsity-inducing methods are powered by some combination of solution truncation and L1 norm minimization.

Passive-aggressive online learning. Within the family of online learning algorithms, a *passive-aggressive* algorithm is one whose update rule *makes no update* to $\mathbf{w}_t$ if $(\boldsymbol{x}_t, y_t)$ is classified correctly with sufficient margin. That is, we *passively* leave $\mathbf{w}_{t+1} = \mathbf{w}_t$. If a margin error occurs, we *aggressively* update $\mathbf{w}_{t+1}$ such that the error is fully correct (though a regularization penalty C will tamper the degree of aggressiveness). Usually, correct classification with sufficient margin is defined using the *hinge loss* ℓ — the algorithm remains passive at timestep t if $\ell\left(\mathbf{w}_t; (\boldsymbol{x}_t, y_t)\right) = \max\left(1 - y_t \mathbf{w}_t^\top \boldsymbol{x}_t, 0\right) = 0$. Next, we introduce the two passive-aggressive algorithms that we will use to test our WRS-Augmented Training method.

Passive-Aggressive Classifier (PAC). Introduced by Crammer et al., PAC is actually a family of three algorithms: PA, PA-I, and PA-II [5]. The base PA algorithm update rule seeks to solve the following constrained optimization problem:

$$\mathbf{w}_{t+1} = \operatorname*{arg\,min}_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 \ \ \text{s.t.} \ \ \ell(\mathbf{w}; (\boldsymbol{x}_t, y_t)) = 0.$$

Because of the hard constraint of forcing $\mathbf{w}_{t+1}$ to satisfy $\ell(\mathbf{w}; (\boldsymbol{x}_t, y_t)) = 0$, the optimization is particularly vulnerable to noisy data. As such, Crammer et al. introduce a new constrained optimization function with a slackness hyperparameter C_{err} to allow for some residual hinge loss and induce less aggressive, but presumably more stable updates:

$$\mathbf{w}_{t+1} = \operatorname*{arg\,min}_{\mathbf{w} \in \mathbb{R}^D} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + C_{err}\epsilon^m \ \ \text{s.t.} \ \ \ell(\mathbf{w}; (\boldsymbol{x}_t, y_t)) \leq \epsilon \ \ \text{and} \ \ \epsilon \geq 0,$$

where setting $m = 1$ corresponds to PA-I and $m = 2$ to PA-II. From initial testing, PA-I and PA-II performed very similarly, with a slight edge to PA-II. As such, for this paper, we will focus on PA-II, which performs the following closed-form update when in aggressive mode [5]: $\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\ell(\mathbf{w}_t; (\boldsymbol{x}_t, y_t))}{\|\boldsymbol{x}_t\|^2 + \frac{1}{2C_{err}}} y_t \boldsymbol{x}_t$. For the remainder of this paper, "PAC" will refer to PA-II.

First-Order Sparse Online Learning (FSOL). Introduced by Zhao et al. [1], FSOL is a passive-aggressive algorithm which attempts to find sparse solutions for $\mathbf{w}_t$. Governed by a learning rate η and a sparsity parameter λ, FSOL keeps track of two vectors $\boldsymbol{\theta}_t, \mathbf{w}_t \in \mathbb{R}^D$ and performs the following update rules when in aggressive mode [1]:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta y_t \boldsymbol{x}_t; \ \ \mathbf{w}_{t+1} = \operatorname{sign}(\boldsymbol{\theta}_{t+1}) \odot [|\boldsymbol{\theta}_{t+1}| - \lambda_t]_+,$$

where $\lambda_t = \eta\lambda$ and $[\boldsymbol{v}]_+$ takes the maximum of each element in $\boldsymbol{v}$ and 0. Zhao et al. note that the above update rules are identical to that of Xiao's RDA method with soft 1-norm regularization [17, 1].

Weighted Reservoir Sampling (WRS). Suppose we have a collection of items $V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T\}$, with corresponding nonnegative weights $w_1, \ldots, w_T$. Our goal is to collect a size-K weighted random sample from V *in one pass* (imagine this process is indexed by time), where the population size $T = |V|$ is potentially unknown. Introduced by Efraimidis and Spirakis [4], *weighted random sampling with a reservoir*, which we shorten to *weighted reservoir sampling* (WRS), is an algorithm that allows us to collect such a size-K weighted random sample under the aforementioned conditions. Specifically, as we are making our one pass through the items in V, at each timestep t, we maintain and update a *reservoir* — a temporary storage unit with a maximum capacity of K items, with each item in the reservoir a potential candidate for our final size-K sample. At time T, the K items that are currently in the reservoir will constitute our sample of size-K. We invite the interested reader to look at Algorithm A in [4] for specific details.

Online-to-batch conversion. Online learning algorithms such as PAC generally do not impose any restrictions on the distribution of the training data sequence. Their *regret* bounds aim to control

3

the cumulative loss M_T of the algorithm over any sequence of data, compared with a minimal-loss fixed model $\hat{\mathbf{w}}$: $\text{reg}_T := \sum_{t=1}^{T} \ell(\mathbf{w}_t; z_t) - \sum_{t=1}^{T} \ell(\hat{\mathbf{w}}; z_t) = M_T - \sum_{t=1}^{T} \ell(\hat{\mathbf{w}}; z_t)$. When using a fixed model to classify unseen data, we need to impose an i.i.d. assumption on the data stream in order to measure the *risk*, or expected generalization error, of the model. Note that the distribution $\mathcal{D}$ itself is arbitrary and can still permit outliers or mixtures. Then the population *risk* is defined as $R_{\mathcal{D}}(\mathbf{w}) := \mathbb{E}_{z \sim \mathcal{D}}[\ell(\mathbf{w}; z)]$.

To theoretically describe our algorithm, we leverage work on online-to-batch conversion, which takes an online algorithm with known regret bounds on an i.i.d. sequence of data and extracts a stable final model with low risk. For example, in the online perceptron algorithm, earlier work studied the *pocket* approach, which selects the longest-surviving model in the sequence as the final model [18, 19]. Other well-known approaches use the uniform average of the whole model sequence or the best-performing model over a validation set [20].

As will be seen, our method generalizes these approaches to utilize multiple long-survival models as an ensemble model. Furthermore, we will introduce novel improvements, including a limited-size reservoir with probabilistic sampling. The risk bounds for our WAT model leverage improved techniques from [21, 22]. Our experimental results also demonstrate that our novel conversion technique outperforms prior baselines (see Appendix D).

3 Our method: WRS-Augmented Training (WAT)

In one extreme, if a candidate solution $\mathbf{w}_t$ from a passive-aggressive algorithm had *perfect classification with sufficient margin* on any given data point, then the *subsequent number of passive steps* taken after time t (i.e., number of timesteps that our algorithm is in passive mode before going aggressive again) would be infinite. In the other extreme, if a candidate solution $\mathbf{w}_t$ had extremely low performance, then our passive-aggressive algorithm is likely to go aggressive very soon after time t, implying a very small subsequent number of passive steps after time t. In short, our key insight is that high-performing solutions $\mathbf{w}_t$ tend to be error-free for more iterations than low-performing solutions. As such, the subsequent number of passive steps taken after the formation of $\mathbf{w}_t$ provides an estimate of $\mathbf{w}_t$'s relative quality (i.e., test accuracy).

However, we do not want to take the intermediate solution $\mathbf{w}_t$ that had the most passive updates as this, too, can be noise (and luck) sensitive. Ideally, we would like to sample from the merging distribution of $\mathbf{w}_t$ as they occur, and take an average of those solutions to obtain a singular, highly robust, solution vector that performs well with little variance. But, we do not wish to store all $\mathbf{w}_t$ due to intractability.

Putting these thoughts together, our WRS-Augmented Training (WAT) method functions as follows. Given a base passive-aggressive algorithm (e.g., PAC or FSOL), we will run said algorithm through our data stream $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$, as normal, but keep a size-K reservoir of promising candidate solutions. The reservoir approach allows us to run through our data stream and collect a weighted random sample of candidate solutions of size-K, *weighted by their subsequent number of passive steps* and without storing all intermediate solutions.

Naturally, this setup is suited for Efraimidis and Spirakis's WRS algorithm. Procedurally, every timestep that our algorithm goes aggressive, we obtain a new active candidate solution. Right before we apply our aggressive mode update rule, we will add the outgoing candidate solution to our size-K reservoir (and remove a current resident of the reservoir, if necessary) following the steps of the WRS algorithm. At any timestep t, we can form an ensemble solution $\mathbf{w}_{\text{WRS}}$ by taking an average of the candidate solutions currently in our reservoir. Hopefully, at any timestep t, $\mathbf{w}_{\text{WRS}}$ will have more stable test performance than the current active candidate solution $\mathbf{w}_t$.

3.1 WAT variants

We will experiment with a few additional variants of the WAT method. First, instead of weighting using the subsequent number of passive steps (which we denote as *standard weights*), what if we weight using the exponentiated subsequent number of passive steps (which we denote as *exponential weights*)? The idea with exponential weights is that it is closer in practice to deterministically picking the candidate solutions with the largest number of subsequent passive steps, while still maintaining some stochasticity — i.e., a "greedier" policy. Second, when constructing our ensemble solution

4

Algorithm 1 WRS-Augmented Training (WAT)

Input: $\mathbf{w}$ - initial solution vector, $\{(\boldsymbol{x}_t, y_t)\}_{t=1}^T$ - data stream, K - reservoir size, WS - weighting scheme ("Standard" or "Exponential"), AS - averaging scheme ("Simple Average" or "Weighted Average"), VZ - voting-based zeroing (True or False), and other base-method-specific hyperparameters $\mathcal{H}$ (e.g., for PAC or FSOL)

Output: WRS-stabilized solution vector $\mathbf{w}_{\text{WRS}}$.

1: **function** WAT($\mathbf{w}, \{(\boldsymbol{x}_t, y_t)\}_{t=1}^T; K, \text{WS}, \text{AS}, \text{VZ}, \mathcal{H}$)
2:      # Initializing intermediate data structures
3:      $s \rightarrow 0$ ▷ Counter for subsequent passive steps of current solution candidate
4:      $\mathcal{R} \rightarrow []$ ▷ Size-K reservoir for storing promising solutions, as array
5:      $\boldsymbol{b}, \mathbf{k} \rightarrow [], []$ ▷ Size-K arrays for weights b_r and auxiliary k_r values for solutions in $\mathcal{R}$
6:      # At each timestep, we observe $(\boldsymbol{x}_t, y_t)$
7:      **for** $t \leftarrow 1$ **to** T **do**
8:          **if** $\ell(\mathbf{w}; \boldsymbol{x}_t, y_t) > 0$ **then**   ▷ If made error, in aggressive mode
9:              # Terminate current solution, probabilistically add to reservoir using WRS [4]
10:              Draw $u^* \sim \text{Unif}(0, 1)$
11:              **if** WS $==$ "Standard" **then**
12:                  $b^* \leftarrow s$
13:              **else if** WS $==$ "Exponential" **then**
14:                  $b^* \leftarrow \exp(s)$
15:              $k^* \leftarrow (u^*)^{\frac{1}{b^* + \epsilon}}$ ▷ $\epsilon = 10^{-8}$ to prevent division by 0
16:              $\tau \leftarrow \min_{j \in 1, \ldots, K} \mathbf{k}[j]$; $i \leftarrow \arg\min_{j \in 1, \ldots, K} \mathbf{k}[j]$
17:              **if** $k^* > \tau$ or $\mathcal{R}$ is not full with K solutions **then**
18:                  $\mathcal{R}[i] \leftarrow \mathbf{w}, \boldsymbol{b}[i] \leftarrow b^*, \mathbf{k}[i] \leftarrow k^*$
19:              # Base method update rule
20:              $\mathbf{w} \leftarrow \mathbf{w} + g(\ldots)$ ▷ $g(\ldots)$ specific to base algorithm (e.g., PAC or FSOL)
21:              $s \leftarrow 0$ ▷ Reset number of subsequent passive steps
22:          **else**   ▷ correctly-classified, still in passive mode
23:              $s \leftarrow s + 1$ ▷ Increment number of subsequent passive steps
24:      # Forming our PAC-WRS solution
25:      **if** AS $==$ "Simple Average" **then**
26:          $\mathbf{w}_{\text{WRS}} \leftarrow \frac{1}{K} \sum_{j=1}^K \mathcal{R}[j]$ ▷ Simple average of solutions in reservoir
27:      **else if** AS $==$ "Weighted Average" **then**
28:          $\mathbf{w}_{\text{WRS}} \leftarrow \sum_{j=1}^K \mathbf{b}[j]\mathcal{R}[j]/(\sum_{j=1}^K \mathbf{b}[j])$ ▷ Weighted avg. of solutions in reservoir
29:      # Voting-based zeroing for extra sparsity
30:      **if** VZ is True **then**
31:          Zero out entries in $\mathbf{w}_{\text{WRS}}$ where the majority of solutions $\mathcal{R}[j]$ contain zeroes.
32:      **Return** $\mathbf{w}_{\text{WRS}}$.

$\mathbf{w}_{\text{WRS}}$, should we take a *simple average* of the residents in our reservoir or a *weighted average*? Third, there are reasonable concerns that constructing $\mathbf{w}_{\text{WRS}}$ via averaging might negate the sparsity advantages of a method like FSOL, due to different candidates in the reservoir containing 0s in different entries. However, if the majority of candidate solutions in the reservoir contain 0s at a given entry, what if we tried zeroing out said entry in $\mathbf{w}_{\text{WRS}}$, as it is likely to be uninformative? We denote this add-on as *voting-based zeroing*.

We present the WAT method in full detail in Algorithm 1. To clarify, the elements in the data structures $\mathcal{R}$, $\mathbf{b}$, and $\mathbf{k}$ (all of which are size-K arrays) are paired with each other, so that when we add/remove an element in $\mathcal{R}$, the corresponding elements in $\mathbf{b}$ and $\mathbf{k}$ are removed as well. To be fully clear, $\mathcal{R}$ contains candidate solution vectors $\mathcal{R}[1], \ldots, \mathcal{R}[K]$, with each $\mathcal{R}[k] \in \mathbb{R}^D$. The vector $\boldsymbol{b}$ contains scalar values $\boldsymbol{b}[1], \ldots, \boldsymbol{b}[K]$, and likewise for the vector $\mathbf{k}$.

3.2 Theoretical Analysis

One of the goals of our method is to choose a set of effective models as the algorithm runs live, without the need for expensive evaluation on a validation set. In this section we first provide *validity*

5

arguments for using observed survival as a proxy to select high-accuracy models. These are finite-sample bounds based on reasoning about the reservoir $\mathcal{R}$. Note that we are interested in studying generalization to unseen data, a complementary setting to prior work which shows convergence in terms of training set error for the perceptron model [19]. After establishing validity, we will turn to learning bounds for our ensemble model risk in the i.i.d. setting. Proofs are deferred to Appendix A.

Validity analysis. We assume a given dataset $z_1^T = \{z_t\}_{t=1}^T$ is an i.i.d. sequence sampled from a generating distribution $\mathcal{D}$. We first suppose that $\ell(\mathbf{w}; z)$ is the 0-1 loss, that is $\ell(\mathbf{w}; (x, y)) = \mathbb{1}(\hat{y} \neq y)$. Then the risk $R_{\mathcal{D}}(\mathbf{w})$ is the probability that a random $z \sim \mathcal{D}$ is misclassified by $\mathbf{w}$. At any time t, we have an online model $\mathbf{w}_t$, for which we define the *survival time* $s_{\mathbf{w}_t}$ to be the number of subsequent correct classifications, stopping when $\mathbf{w}_t$ misclassifies a sample. As $\mathbf{w}_t$ does not change until an error occurs, at any finite time we only collect K updated models $\{\mathbf{w}^{(j)}\}_{j=1}^K$ into $\mathcal{R}$.

Our first result bounds the difference in risk among two models in $\mathcal{R}$: (1) the minimal-risk hypothesis $\tilde{\mathbf{w}} = \arg \min R_{\mathcal{D}}(\mathbf{w}^{(j)})$ which we do not know, and (2) the longest-surviving hypothesis $\mathbf{w}_s = \arg \min s_{\mathbf{w}^{(j)}}$ which we observe. We also denote $s_{\tilde{\mathbf{w}}}$ and $s_{\mathbf{w}_s}$ to be their respective survival times.

Theorem 1. *Let $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)} \in \mathcal{R}$ be the updated outputs of an online PA algorithm on inputs $Z_1^T \sim \mathcal{D}$. Also define r_m as the minimal achievable risk of any model, such that $r_m \leq R_{\mathcal{D}}(\tilde{\mathbf{w}})$ almost surely. Then*

$$P(R_{\mathcal{D}}(\mathbf{w}_s) > R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon) \leq \min \left\{ K \frac{r_m}{2r_m + \varepsilon - r_m(r_m + \varepsilon)}, e^{-r_m^K(r_m + \varepsilon)} \right\}$$

This immediately leads to

Corollary 1. *With prob. $1 - \delta$, $R_{\mathcal{D}}(\mathbf{w}_s) \leq R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \min \left\{ \frac{1 - \delta - r_m^{K+1}}{r_m^K}, \frac{r_m(K - \delta(2 - r_m))}{\delta(1 - r_m)} \right\}$.*

While this bound explains the use of top-1 survival in the worst case, we further justify the use of an ensemble (or reservoir) of top-surviving models. Specifically, given the use of top-B models, there is always a certain probability that the top-$(B + 1)$ can lower the risk. Since $B = 1$ is a base case, this implies that any value of B is worthy of consideration, until this probability decays to 0.

To simplify the notation, we sort the models in $\mathcal{R}$ by decreasing survival time $s'_1, \ldots, s'_k$, with corresponding re-indexed weights $w'_1, \ldots, w'_k$ and risks $R'_1, \ldots, R'_k$. We also add an assumption, that there is a partition of $\mathcal{R} = \mathcal{R}_g \cup \mathcal{R}_b$ such that $\mathcal{R}_g = \{\mathbf{w} \in \mathcal{R} : r_m \leq R_{\mathcal{D}}(\mathbf{w}) \leq r_m + \varepsilon\}$ and $\mathcal{R}_b = \{\mathbf{w} \in \mathcal{R} : r_m + \varepsilon < R_{\mathcal{D}}(\mathbf{w}) \leq r_m + 3\varepsilon\}$, and that $|\mathcal{R}_b| \geq B$. That is, there is a set of *good* and *bad* models in terms of risk. In the Appendix we show that the assumption is readily satisfied.

Theorem 2. *Let $\bar{R}_{\mathcal{D}}(\mathbf{w}_B)$ be the averaged risk of the top-B surviving models, and let $\bar{R}_{\mathcal{D}}(\mathbf{w}_{B+1})$ be the averaged risk including the next highest survival model. Then $\bar{R}_{\mathcal{D}}(\mathbf{w}_{B+1}) \leq \bar{R}_{\mathcal{D}}(\mathbf{w}_B)$ with probability at least $|\mathcal{R}_g| \binom{|\mathcal{R}_b|}{B} r_m^2 (r_m + 3\varepsilon)^{|\mathcal{R}_b| - B} (1 - r_m - 3\varepsilon)^B$.*

Finally note that we can define an averaged model (weighted or unweighted) $\bar{\mathbf{w}}_B$. For any convex $\ell_h(\mathbf{w}; z)$ with risk $R_{\mathcal{D}}^h$, such as the hinge loss, Jensen's inequality gives $R_{\mathcal{D}}^h(\bar{\mathbf{w}}_{B+1}) \leq \bar{R}_{\mathcal{D}}(\mathbf{w}_{B+1})$.

Learning bounds. Now we turn to generalization bounds of our method, when run to a fixed stopping time T. We assume more generally that $\ell(\mathbf{w}; z)$ is convex, such as the hinge loss. Furthermore we suppose that the loss of any point in the training set is bounded by C. This is a safe assumption for many passive-aggressive algorithms, where the input data is normalized and the update steps are not too large.

Theorem 3. *Suppose ℓ is convex and bounded from above by C. By time T, suppose the reservoir contains K_T models, with survival time of each at least s_T. Let $M_{wrs} = \sum_{t=1}^T \pi_t \ell(\mathbf{w}_{t-1}; z_t)$ be the cumulative loss of the WAT reservoir sequence, with π_t formally defined in the proof. Then w.p. $1 - \delta$,*

$$R_{\mathcal{D}}(\mathbf{w}_{wrs}) \leq \frac{M_{wrs}}{K_T s_T} + \sqrt{\frac{2C \log\left(\frac{T}{\delta}\right) M_{wrs}}{(K_T s_T)^2}} + \frac{7C \log\left(\frac{T}{\delta}\right)}{K_T s_T}$$

The result shows that the risk of the ensemble model is stochastically bounded by the cumulative loss of the online procedure. By applying known regret bounds for the underlying online algorithms (e.g. PAC or FSOL), we can further bound the risk in terms of the original learner, and subsequently

6

the optimal risk. In particular, we can "abstract out" the actual online learning method in the proof, as a generic regret bound of form $\frac{M_T}{T} \leq \frac{1}{T} \sum_{t=1}^{T} \ell(\mathbf{w}; z_t) + \frac{r(T)}{T}$, for any $\mathbf{w}$. Depending on the algorithm often $r(T) = O(\sqrt{T})$ or even $O(1)$.

With this in hand, we can show that the risk actually approaches the minimal risk. One additional assumption is required: $M_{wrs} / \sum_{t=1}^{T} \pi_t \leq \frac{1}{T} \sum_t \ell(\mathbf{w}_{t-1}; z_t) = \frac{M_T}{T}$ (the WRS sequence has lower cumulative loss than the original learner). By definition, the reservoir contains the models which have longest survivals, and hence lowest regret density. So this statement is readily satisfied, as long as the distributions of non-zero losses are not different among the two sequences. In fact, under certain conditions, the inequality is likely strict which further improves the bound. Altogether we obtain:

Theorem 4. *Given a PA algorithm, let M_T be its cumulative loss, and $r(T)$ be the algorithm-specific excess regret. Then with probability $1 - \delta$, the deviation in risk of our WRS model $\mathbf{w}_{wrs}$ from the optimal model $\mathbf{w}^*$ is bounded as*

$$R_{\mathcal{D}}(\mathbf{w}_{wrs}) \leq R_{\mathcal{D}}(\mathbf{w}^*) + \frac{r(T)}{T} + \sqrt{\frac{2C \log\left(\frac{2T}{\delta}\right) M_T}{T K_T s_T} + \frac{7C \log\left(\frac{2T}{\delta}\right)}{K_T s_T}} + C\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2T}}$$

Table 1: Sizes, dimensions, and sparsities of all datasets used for numerical experiments.

Dataset	D	N	$\mathbf{N_{train}}$	$\mathbf{N_{test}}$	Sparsity
Avazu (App) [23] (via LIBSVM)	1000000	14596137	10217295	4378842	0.999985
Avazu (Site) [23] (via LIBSVM)	1000000	25832830	18082981	7749849	0.999985
Criteo (via LIBSVM)	1000000	51882752	36317926	15564826	0.999961
Dexter [24]	20000	600	420	180	0.995319
Dorothea [25]	100000	1150	805	345	0.990909
KDD2010 (Algebra) [26] (via LIBSVM)	20216830	8918054	6242637	2675417	0.999998
MNIST8 (4+9) [27, 28] (via LIBSVM)	784	1591785	1114249	477536	0.757170
News20 [29] (via LIBSVM)	1355191	19954	13967	5987	0.999664
Newsgroups (Binary, CS) [30, 31] (via sklearn)	101631	18311	12817	5494	0.999049
PCMAC [32]	3289	1943	1360	583	0.985418
RCV1 [33] (via LIBSVM)	47236	697641	488348	209293	0.998451
Real-Sim (McCallum via LIBSVM)	20958	72201	50540	21661	0.997549
SST-2 [34]	13757	67337	47135	20202	0.999421
URL [35] (via LIBSVM)	3231961	2396130	1677291	718839	0.999964
W8A [36] (via LIBSVM)	300	59245	41471	17774	0.957585
Webspam [37] (via LIBSVM)	254	350000	244999	105001	0.664833

4   Numerical experiments

We combine WAT with base PAC and FSOL, forming two new methods PAC-WRS and FSOL-WRS[1]. We evaluate their performances across 16 binary classification datasets listed in Table 1. Please see Appendix B for more dataset details. We are interested in three metrics: **1) Final test accuracy:** proportion of test set data points correctly-classified by solution obtained after making one pass through the training data. **2) Final sparsity:** proportion of zeroes in our classification solution obtained after making one pass through the training data. **3) Relative oracle performance (ROP):** let $p_{t,\text{base}}$ be the test accuracy of our base model (either PAC or FSOL) at time t and p_t be the test accuracy of our model of interest (either base PAC, base FSOL, PAC-WRS, or FSOL-WRS). Then, define $p_{t,\text{oracle}} = \max_{\text{base}} p_{t,\text{base}}$ to be the cumulative maximum test accuracy of the base method at time t. In other words, $p_{t,\text{oracle}}$ represents the highest performance we could obtain if we had an oracle telling us which candidate solution vector we encountered was best. Then, define ROP $= \frac{1}{T} \sum_{t=1}^{\top} (p_{t,\text{oracle}} - p_t)$. Intuitively, if a method has very stable test accuracy over time with minimal fluctuations, then ROP should be close to 0, or negative (i.e. achieving a higher test accuracy than the oracle). In contrast, a large, positive ROP suggests fluctuations in test accuracy.

Because we are interested in the stability of WAT's test-set accuracy over the entirety of the training run, it is most illustrative to look at figures when possible. Representative figures will be shown in the

---

[1]Code available at `https://github.com/FutureComputing4AI/Weighted-Reservoir-Sampling-Augmented-Training`

7

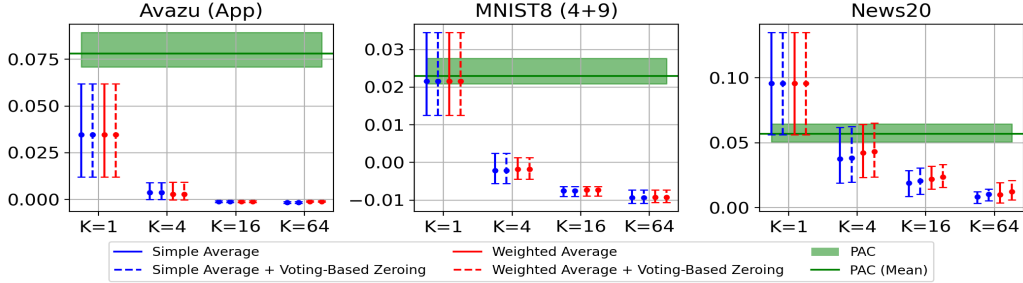



Figure 2: Relative oracle performances (y-axis) of base PAC and PAC-WRS using standard weights over reservoir sizes K (x-axis) on 3 representative datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity. **Lower values indicate more stable performance.**

main paper, with all figures for all datasets in the appendix. When running any algorithm (PAC, FSOL, PAC-WRS, or FSOL-WRS) on any dataset, we perform a random 70/30 train-test split. We begin by tuning the C_{err}, η, and λ hyperparameters for base PAC and FSOL, with details located in Appendix B. For PAC-WRS and FSOL-WRS, we use the hyperparameters for the corresponding base models, but try all possible WAT variants of weighting scheme (standard or exponential), averaging scheme (simple vs. weighted), voting-based zeroing (True or False), and reservoir size $K \in \{1, 4, 16, 64\}$. We perform five trials for each PAC-WRS and FSOL-WRS variant with randomized shuffles of the training and test data, running through the training data only once for each trial. All candidate solutions are initialized as the **0** vector. Please see more experimental details in Appendix B.

4.1 Stabilizing test performance

Figure 1 shows visually how PAC-WRS and FSOL-WRS are highly-effective at stabilizing base PAC and FSOL's wildly-fluctuating test accuracy on Avazu (App) and News20. Corresponding figures for all 16 datasets and PAC-WRS/FSOL-WRS variants can be found in Appendix G.2. From Figure 2, we see that PAC-WRS is highly-effective at reducing ROP compared to base PAC, and that **the larger the reservoir size K, the more stable the resultant test accuracy for $\mathbf{w_{WRS}}$ becomes.** Furthermore, looking more carefully at MNIST8 (4+9), we observe that many PAC-WRS variants were able to achieve *negative* ROP values, suggesting that $\mathbf{w}_{WRS}$ could achieve *higher test accuracies* than even the oracle. Corresponding figures for all 16 datasets can be found in Appendix G.1.

From Table 2, we observe that FSOL-WRS with standard weights and simple averaging **successfully stabilized test accuracy in all 16 tested datasets compared to base FSOL**, as measured via ROP.

PAC-WRS with exponential weights and simple averaging successfully stabilized test accuracy in 14 of 16 tested datasets compared to base PAC. Following the best practices in [38, 39], we perform Wilcoxon signed-rank tests for sta-

Table 2: Numbers of datasets out of 16 where each PAC-WRS or FSOL-WRS variant with $K = 64$ outperformed its corresponding base method (PAC or FSOL), as measured by ROP averaged across 5 randomized trials.

	Simple Average		Weighted Average	
	Standard	Exp.	Standard	Exponential
PAC	13	14	13	11
FSOL	16	13	16	11

tistical significance on the differences in ROP between PAC/FSOL-WRS versus base PAC and FSOL, taking into account performance on all 16 datasets. At a significance level of $\alpha = 0.05$, we find that both PAC/FSOL-WRS achieve statistically-significant reductions in ROP compared to their corresponding base models when equipped with standard weights ($p < 0.0386$ in each case, see Table 6 in Appendix F).

These results also suggest that standard weights are preferable to exponential weights. This makes sense because using exponential weights may be too "greedy," causing the algorithm to overly trust in the number of passive steps as an indicator of test performance, polluting the reservoir with poor "lucky" candidate solutions, and refusing to remove them later.

8

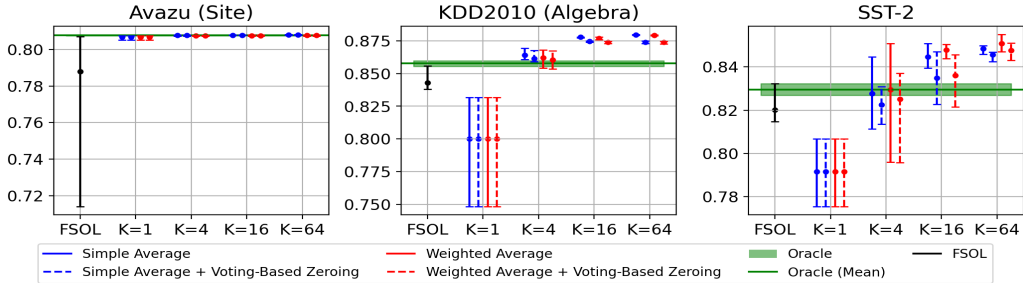



Figure 3: Final test accuracies (y-axis) of base FSOL and FSOL-WRS using standard weights over reservoir sizes K (x-axis) on 3 representative datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. See Figure 2 for legend description.

A note on final test accuracy. From Figure 3, we see that the final test accuracies achieved by FSOL-WRS are not only higher on average than base FSOL, but also have significantly lower variance. On these datasets, there is no significant difference between FSOL-WRS variants with or without voting-based zeroing. As expected, final test accuracy seems to be higher for larger values of reservoir size K. However, Wilcoxon signed-rank tests to compare PAC-WRS and FSOL-WRS's improvements in final test accuracy over base PAC and FSOL indicate that FSOL-WRS with standard weights yields a statistically-significant improvement in final test accuracy compared to base FSOL, but PAC-WRS does not compared to base PAC (see Table 7 in Appendix F). One hypothesis for this discrepancy is that it is relatively unlikely for any given training data point to cause a massive drop in test accuracy. Furthermore, subsequent data points will usually help the base model self-correct (see Appendix G.2). Thus, this explains why the base method's mean final test accuracy after a particular fixed timestep (e.g., N_{train}) will not differ significantly from that of the WRS-augmented method, especially not across only 5 trials.

Nonetheless, there are enough data points in the training stream that these massive fluctuations in test accuracy could still happen thousands of times throughout the training process, corroborating what we saw in Figure 1. If such a drop in test accuracy were to occur at an unlucky timestep when the model training is stopped, the consequences could be unacceptable. Furthermore, in continuously updated "any-time" environments, it is hard to assess if one of these drops has occurred. As such, WAT is valuable as a simple and effective way of preventing such fluctuations in test accuracy.

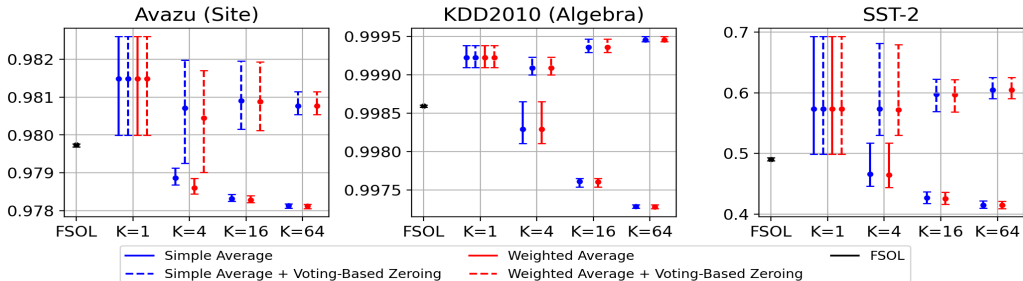



Figure 4: Final sparsities (y-axis) of base FSOL and FSOL-WRS using standard weights over reservoir sizes K (x-axis) on 3 representative datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. See Figure 2 for legend description.

Preservation of sparsity. Finally, we aim to retain the favorable weight sparsity from FSOL. From Figure 4, we observe that even with reservoir size $K = 64$, the final sparsities achieved by FSOL-WRS are quite similar to that of base FSOL. Furthermore, we observe that with voting-based zeroing, FSOL-WRS often achieves even higher sparsity than the base model. Please see Appendix G.1 for additional

Table 3: Numbers of datasets out of 16 where each top-64 PAC or FSOL variant outperformed its corresponding base method (PAC or FSOL), as measured by relative oracle performance averaged across 5 trials.

	Simple Average	Weighted Average
PAC	10	12
FSOL	15	15

9

errorbar-type plots and Appendix G.2 for sparsity over timestep. One explanation for why even FSOL-WRS variants *without* voting-based zeroing can achieve similar sparsity to base FSOL is that the final reservoir will likely contain candidate solutions from significantly earlier timesteps, when weights were sparser due to fewer aggressive updates. Thus, WAT can maintain sparsity despite the use of weight averaging.

4.2 Ablations and extensions

Top-K ablation. A natural question that one might ask is — instead of probabilistically sampling candidate solutions, what if we deterministically picked the top-K candidate solutions with the largest subsequent numbers of passive steps and took their simple and/or weighted average? Setting $K = 64$, we see from Table 3 that the best top-64 PAC/FSOL variants are not as effective as stabilizing test accuracy as the best PAC/FSOL-WRS variants, as shown in Table 2 and measured via ROP. Furthermore, Wilcoxon signed-rank test p-values from Table 4 also suggest that this top-64 ablation cannot produce as statistically-significant increases in test accuracy stability compared to WAT, as measured via ROP. Like WAT, top-K is also ineffective at producing statistically-significant increases in final test accuracy on PAC. On FSOL, top-K can produce statistically significant increases in final test accuracy, but these p-values are over an order of magnitude larger than FSOL-WRS's (see Table 7 in Appendix F). As such, WAT is still the best for stabilizing test accuracy.

Table 4: Wilcoxon signed-rank test p-values testing whether differences in **relative oracle performance** and **final test accuracy** between top-64 PAC/FSOL variants and base PAC/FSOL are statistically significant.

	Relative Oracle Performance		Final Test Accuracy	
	Simple Average	Weighted Average	Simple Average	Weighted Average
PAC	0.162	0.0298	0.214	0.255
FSOL	0.00270	0.00270	0.0130	0.0130

Additional ablations and comparisons. For the interested reader, in Appendix D we include empirical performance comparisons of WRS-Augmented Training against two traditional ensembling mechanisms: moving average (e.g., averaging the most-recently-observed $K = 64$ weight vectors at each timestep) and exponential average (e.g., forming an ensemble vector $\bar{\mathbf{w}}_t = \gamma \mathbf{w}_t + (1-\gamma)\bar{\mathbf{w}}_{t-1}$ at each timestep), where $\mathbf{w}_t$ is the base algorithm's candidate solution at timestep t. In short, especially in more real-world, large-scale settings where evaluation and checking are prohibitively expensive, WRS-Augmented Training is the fastest, most accurate, and most reliable method compared to all the aforementioned baselines.

Modifying WAT for non-passive aggressive methods. While the main theoretical and empirical results in this paper were primarily oriented towards passive-aggressive base models, in Appendix E, we include empirical simulations of applying a modified form of WRS-Augmented Training on top of three non-passive-aggressive online learning methods: Stochastic Gradient Descent with Momentum [40], ADAGRAD [14], and Truncated Gradient Descent [15]. In general, our modified WRS-Augmented Training effectively mitigates test accuracy when it exists, and does minimal harm when it does not.

5 Conclusion, limitations, and future work

In this paper, we introduced WRS-Augmented Training (WAT), a procedure that can be used to stabilize any passive-aggressive online learning algorithm, neither requiring a hold-out evaluation set nor additional passes over the training data. We applied WAT to base PAC and FSOL, demonstrating across 16 datasets that WAT is highly effective at mitigating the massive fluctuations in test accuracy between timesteps that affect many online learning algorithms. WAT runs at minimal cost, with only a fixed K multiple on memory for multiple weight vectors to be saved.

One limitation of this work is that WAT implicitly assumes that the training and test data come from fixed distributions that do not change over time. However, some data distributions will evolve over time. As such, a candidate solution that entered the reservoir early on due to having a large number of subsequent passive steps might not actually retain its performance as the data distribution evolves over time. While they may eventually be replaced, non-IID adaptions may be useful in the future.

10

References

[1] Peilin Zhao, Dayong Wang, Pengcheng Wu, and Steven CH Hoi. A Unified Framework for Sparse Online Learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(5):1–20, 2020.

[2] Nikos Karampatziakis and John Langford. Online Importance Weight Aware Updates, 2011.

[3] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. Moa: Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the first workshop on applications of pattern analysis*, pages 44–50. PMLR, 2010.

[4] Pavlos S Efraimidis and Paul G Spirakis. Weighted Random Sampling with a Reservoir. *Information Processing Letters*, 97(5):181–185, 2006.

[5] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7(19):551–585, 2006.

[6] Edward Raff, Bobby Filar, and James Holt. Getting Passive Aggressive About False Positives: Patching Deployed Malware Detectors. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 506–515. IEEE, nov 2020.

[7] H. S. Anderson and P. Roth. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *ArXiv e-prints*, April 2018.

[8] Richard Harang and Ethan M. Rudd. Sorel-20m: A large scale benchmark dataset for malicious pe detection. In *Proceedings of the Conference on Applied Machine Learning for Information Security*, 2021.

[9] Robert J Joyce, Edward Raff, Charles Nicholas, and James Holt. MalDICT: Benchmark Datasets on Malware Behaviors, Platforms, Exploitation, and Packers. *Proceedings of the Conference on Applied Machine Learning in Information Security*, 2023.

[10] Robert J Joyce, Edward Raff, and Charles Nicholas. A Framework for Cluster and Classifier Evaluation in the Absence of Reference Labels. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security (AISec '21)*. Association for Computing Machinery, 2021.

[11] Robert J Joyce, Edward Raff, and Charles Nicholas. Rank-1 Similarity Matrix Decomposition For Modeling Changes in Antivirus Consensus Through Time. In *Proceedings of the Conference on Applied Machine Learning for Information Security*, 2021.

[12] Robert J. Joyce, Tirth Patel, Charles Nicholas, and Edward Raff. AVScan2Vec: Feature Learning on Antivirus Scan Data for Production-Scale Malware Corpora. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, page 185–196, New York, NY, USA, 2023. Association for Computing Machinery.

[13] Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive Regularization of Weight Vectors. *Machine learning*, 91:155–187, 2013.

[14] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(7), 2011.

[15] John Langford, Lihong Li, and Tong Zhang. Sparse Online Learning via Truncated Gradient. *Journal of Machine Learning Research*, 10(3), 2009.

[16] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic Methods for L1 Regularized Loss Minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 929–936, 2009.

[17] Lin Xiao. Dual Averaging Method for Regularized Stochastic Learning and Online Optimization. *Advances in Neural Information Processing Systems*, 22, 2009.

11

[18] Yoav Freund and Robert E Schapire. Large Margin Classification Using the Perceptron Algorithm. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 209–217, 1998.

[19] Marco Muselli. On convergence properties of pocket algorithm. *IEEE Transactions on Neural Networks*, 8(3):623–629, 1997.

[20] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the Generalization Ability of On-line Learning Algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

[21] Nicolo Cesa-Bianchi and Claudio Gentile. Improved Risk Tail Bounds for On-line Algorithms. *IEEE Transactions on Information Theory*, 54(1):386–390, 2008.

[22] Ofer Dekel. From Online to Batch Learning with Cutoff-Averaging. *Advances in Neural Information Processing Systems*, 21, 2008.

[23] Steve Wang and Will Cukierski. Click-Through Rate Prediction. Kaggle. https://kaggle.com/competitions/avazu-ctr-prediction, 2014.

[24] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Dexter. UCI Machine Learning Repository, 2008. DOI: https://doi.org/10.24432/C5P898.

[25] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, , and Gideon Dror. Dorothea. UCI Machine Learning Repository, 2008. DOI: https://doi.org/10.24432/C5NK6X.

[26] J. Stamper, A. Niculescu-Mizil, S. Ritter, G.J. Gordon, and K.R. Koedinger. Algebra I 2008-2009. Challenge Data Set from KDD Cup 2010 Educational Data Mining Challenge. Find it at http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp, 2010.

[27] Jingwei Liang and Clarice Poon. Screening for Sparse Online Learning. *arXiv preprint arXiv:2101.06982*, 2021.

[28] Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training Invariant Support Vector Machines using Selective Sampling. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

[29] S Sathiya Keerthi, Dennis DeCoste, and Thorsten Joachims. A Modified Finite Newton Method for Fast Solution of Large Scale Linear SVMs. *Journal of Machine Learning Research*, 6(3), 2005.

[30] Ken Lang. Newsweeder: Learning to Filter NetNews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature Selection: A Data Perspective. *ACM Computing Surveys (CSUR)*, 50(6):94, 2018.

[33] David D Lewis, Yiming Yang, Tony Russell-Rose, and Fan Li. Rcv1: A New Benchmark Collection for Text Categorization Research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[34] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[35] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying Suspicious urls: An Application of Large-Scale Online Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 681–688, 2009.

12

[36] John C Platt. Fast Training of Support Vector Machines Using Sequential Minimal Optimization. 1998.

[37] Steve Webb, James Caverlee, and Calton Pu. Introducing the Webb Spam Corpus: Using Email Spam to Identify Web Spam Automatically. In *CEAS*, 2006.

[38] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should We Really Use Post-Hoc Tests Based on Mean-Ranks? *Journal of Machine Learning Research*, 17(5):1–10, 2016.

[39] Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[40] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[41] David A Freedman. On tail probabilities for martingales. *the Annals of Probability*, pages 100–118, 1975.

13

A Proofs

A.1 Validity analysis

Proof of Theorem 1. The main statement involves two independent bounds. See the proofs for Theorem 5 and Theorem 6.

Theorem 5. *Let $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)} \in \mathcal{R}$ be the updated outputs of an online PA algorithm on inputs $z_1^T \sim \mathcal{D}$. Define $\mathbf{w}_s$ and $\tilde{\mathbf{w}}$ as previously. Also define r_m as the minimal achievable risk of any model, such that $r_m \leq R_{\mathcal{D}}(\tilde{\mathbf{w}})$ almost surely.*

Then

$$P(R_{\mathcal{D}}(\mathbf{w}_s) > R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon) \leq K \frac{r_m}{2r_m + \varepsilon - r_m(r_m + \varepsilon)}$$

Proof. First note that while $R_{\mathcal{D}}(\mathbf{w})$ is a constant value for any fixed $\mathbf{w}$, $R_{\mathcal{D}}(\mathbf{w}_s)$ and $R_{\mathcal{D}}(\tilde{\mathbf{w}})$ are random variables induced by the distribution of Z_1^T. The sequence of observed samples affects both the algorithm's output $\mathbf{w}$ as well as the subsequent survival of $\mathbf{w}$.

Next, we define the event of interest $A = \{R_{\mathcal{D}}(\mathbf{w}_s) > R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon\}$. For a given realization of Z_1^T let the set $\mathcal{R}_\varepsilon = \{\mathbf{w} \in \mathcal{R} : R_{\mathcal{D}}(\mathbf{w}) > R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon\}$. Then

$$\begin{aligned} A &\subseteq \{\exists \mathbf{w} \in \mathcal{R} : (s_{\mathbf{w}} \geq s_{\tilde{\mathbf{w}}}) \cap (R_{\mathcal{D}}(\mathbf{w}) > R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon)\} \\ &= \{\exists \mathbf{w} \in \mathcal{R}_\varepsilon : s_{\mathbf{w}} \geq s_{\tilde{\mathbf{w}}}\} \end{aligned}$$

We aim to split this into separate events and apply a union bound, but the contents and size of $\mathcal{R}$ depend on Z_1^T.

Therefore we enumerate the models within each $\mathcal{R}_\varepsilon$ in the order they are received: $\{\mathbf{w}_\varepsilon^{(1)}, \mathbf{w}_\varepsilon^{(2)}, \ldots\}$, up to $\mathbf{w}_\varepsilon^{(K)}$, enabling the index i to refer to the corresponding model $\mathbf{w}_\varepsilon^{(i)}$ in any realization of Z_1^T. Then we can split the event that a hypothesis in the reservoir has longer survival into the union of events on each ordered hypothesis.

We have

$$\begin{aligned} A &\subseteq \{\exists \mathbf{w} \in \mathcal{R}_\varepsilon : s_{\mathbf{w}} \geq s_{\tilde{\mathbf{w}}}\} \\ &\subseteq \bigcup_i \{s_{\mathbf{w}_\varepsilon^{(i)}} \geq s_{\tilde{\mathbf{w}}}\} \end{aligned}$$

Applying the union bound, Lemma 1, and then Lemma 2,

$$\begin{aligned} P(A) &\leq \sum_i P(s_{\mathbf{w}_\varepsilon^{(i)}} \geq s_{\tilde{\mathbf{w}}}) \\ &\leq \sum_i \mathbb{E}_{\mathbf{w}_\varepsilon^{(i)}, \tilde{\mathbf{w}}} P(s_{\mathbf{w}_\varepsilon^{(i)}} \geq s_{\tilde{\mathbf{w}}} | \mathbf{w}_\varepsilon^{(i)}, \tilde{\mathbf{w}}) \\ &= \sum_i \mathbb{E}_{\mathbf{w}_\varepsilon^{(i)}, \tilde{\mathbf{w}}} P(S(\mathbf{w}_\varepsilon^{(i)}) \geq S(\tilde{\mathbf{w}}) | \mathbf{w}_\varepsilon^{(i)}, \tilde{\mathbf{w}}) \\ &= \sum_i \mathbb{E}_{\mathbf{w}_\varepsilon^{(i)}, \tilde{\mathbf{w}}} \frac{R_{\mathcal{D}}(\tilde{\mathbf{w}})}{R_{\mathcal{D}}(\mathbf{w}_\varepsilon^{(i)}) + R_{\mathcal{D}}(\tilde{\mathbf{w}}) - R_{\mathcal{D}}(\mathbf{w}_\varepsilon^{(i)})R_{\mathcal{D}}(\tilde{\mathbf{w}})} \\ &\leq K \frac{r_m}{2r_m + \varepsilon - r_m(r_m + \varepsilon)} \end{aligned}$$

To further clarify, the probabilities are computed over the probability space of $Z_1^T \sim \mathcal{D}$, and $\tilde{\mathbf{w}}$ depends on the realization of Z_1^T. Lemma 1 allows us to reason about the dependent model instances as independent geometric variables, but we need to extract $\tilde{\mathbf{w}}$ into a instance-independent state. Thus we need to condition on the weight values, and then bound the probability uniformly across the expectation, using the premise that the weight risks are ε-distant. $\square$

14

Theorem 6. *(Alternative bound) Let the same conditions hold as in Theorem 5. Then*

$$P(R_{\mathcal{D}}(\mathbf{w}_s) > R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon) \leq e^{-r_m^K(r_m+\varepsilon)}$$

Proof. As before, let $A = \{R_{\mathcal{D}}(\mathbf{w}_s) > R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon\}$. Then

$$
\begin{aligned}
A \subseteq \{R_{\mathcal{D}}(\mathbf{w}_s) > R_{\mathcal{D}}(\tilde{\mathbf{w}})\} &\subseteq \{\mathbf{w}_s \neq \tilde{\mathbf{w}}\} \\
&= \{\mathbf{w}_s = \tilde{\mathbf{w}}\}^c \\
&= \{s_{\tilde{\mathbf{w}}} \geq s_{\mathbf{w}}, \forall \mathbf{w} \in \mathcal{R}\}^c
\end{aligned}
$$

The event $\{\mathbf{w}_s = \tilde{\mathbf{w}}\}^c$ implies $\tilde{\mathbf{w}}$ had or was tied for the highest survival, and we make the simplification that ties are resolved in favor of $\tilde{\mathbf{w}}$. Then

$$
\begin{aligned}
P(A) &\leq 1 - P(s_{\tilde{\mathbf{w}}} \geq s_{\mathbf{w}}, \forall \mathbf{w} \in \mathcal{R}) \\
&= 1 - P(s_{\tilde{\mathbf{w}}} \geq \max_{\mathbf{w}} s_{\mathbf{w}}, \forall \mathbf{w} \in \mathcal{R}) \\
&= 1 - \sum_{x=1}^{\infty} R_{\mathcal{D}}(\tilde{\mathbf{w}})(1 - R_{\mathcal{D}}(\tilde{\mathbf{w}}))^{x-1} \prod_{j=1}^{K} \left(1 - (1 - R_{\mathcal{D}}(\mathbf{w}^{(j)}))^x\right) \\
&\leq 1 - \sum_{x=1}^{\infty} R_{\mathcal{D}}(\tilde{\mathbf{w}})(1 - R_{\mathcal{D}}(\tilde{\mathbf{w}}))^{x-1} \left(1 - (1 - R_{\mathcal{D}}(\tilde{\mathbf{w}})^x\right)^{K-1} \left(1 - (1 - R_{\mathcal{D}}(\tilde{\mathbf{w}}) - \varepsilon)^x\right)
\end{aligned}
$$

where in the third step, we apply Lemma 3. In the last step, we use the premise to suppose that there is at least one hypothesis which is ε-worse than $\tilde{\mathbf{w}}$.

Considering the first-term only,

$$
\begin{aligned}
P(A) &\leq 1 - R_{\mathcal{D}}(\tilde{\mathbf{w}})(R_{\mathcal{D}}(\tilde{\mathbf{w}}))^{K-1}(R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon) \\
&= 1 - R_{\mathcal{D}}(\tilde{\mathbf{w}})^K(R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon) \\
&\leq \exp\left(-R_{\mathcal{D}}(\tilde{\mathbf{w}})^K(R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon)\right) \\
&\leq \exp\left(-r_m^K(r_m + \varepsilon)\right)
\end{aligned}
$$

$\square$

Remark. *Note that this gives the worst-case bound presuming no knowledge over the distribution of risks among the K hypotheses. In such a scenario, the worst case is that all hypotheses achieve $R_{\mathcal{D}}(\tilde{\mathbf{w}})$ except for one which is $R_{\mathcal{D}}(\tilde{\mathbf{w}}) + \varepsilon$. This is unrealistically adversarial, but the bound can easily be tightened by adding assumptions that the risks $R_{\mathcal{D}}(\mathbf{w}^{(j)})$ are well-behaved and applying those in the last step.*

Lemma 1. *Let $R_{\mathcal{D}}(\mathbf{w})$ be the misclassification risk. Suppose an online learner outputs hypothesis $\mathbf{w}_t$ at time t after observing samples Z_1^t. Let s_t be the subsequent survival time of $\mathbf{w}_t$, conditional on the observed sequence Z_1^t. Also let $S(\mathbf{w}_t)$ be the survival of $\mathbf{w}_t$ over the distribution of sample sequences from $\mathcal{D}$. Then $S(\mathbf{w}_t)$ has a Geometric distribution with parameter $R(\mathbf{w}_t)$. In addition $s_t \overset{d}{=} S(\mathbf{w})$.*

Proof. For the first statement, observe that when sampling z independently from $\mathcal{D}$,

$$R(\mathbf{w}_t) = \mathbb{E}_{\mathcal{D}}[\mathbb{1}(\mathbf{w}_t(x) \neq y)] = P_{(x,y) \sim \mathcal{D}}(\mathbf{w}_t(x) \neq y)$$

Since each i.i.d. sample has a probability $R(\mathbf{w}_t)$ to be misclassified, the number of trials until the first misclassification is a Geometric distribution with parameter $R(\mathbf{w}_t)$.

For the second, we need to consider a sequence of correct/incorrect classifications by $\mathbf{w}_t$ as a Bernoulli process of r.v. W_i with parameter $R(\mathbf{w}_t)$, with a stopping time τ indicated by the first misclassification. τ is a random variable and we see that $S_t = \tau - t$. Similarly $S(\mathbf{w}_t)$ can be defined on another Bernoulli process starting from $t = 0$. From the memoryless property, these have the same distribution. $\square$

15

Lemma 2. *Given random variables X and Y which are independent Geometric with parameters p and q respectively (e.g. PMF of X is $P(X = k) = (1-p)^{k-1}p$, then*

$$P(X \geq Y) = \frac{q}{p + q - pq}$$

and

$$P(X > Y) = \frac{q - pq}{p + q - pq}$$

Proof. For the first statement,

$$
\begin{aligned}
P(X \geq Y) &= \sum_{y=1}^{\infty} P(Y = y \cap X \geq y) \\
&= \sum_{y=1}^{\infty} P(X \geq y | Y = y) P(Y = y) \\
&= \sum_{y=1}^{\infty} P(X \geq y) P(Y = y) \\
&= \sum_{y=1}^{\infty} \sum_{x=y}^{\infty} P(X = x) P(Y = y) \\
&= \sum_{y=1}^{\infty} \sum_{x=y}^{\infty} (1-p)^{x-1} p \cdot (1-q)^{y-1} q \\
&= pq \sum_{y=1}^{\infty} (1-q)^{y-1} \sum_{x=y}^{\infty} (1-p)^{x-1} \\
&= pq \cdot \sum_{y=1}^{\infty} (1-q)^{y-1} \cdot \frac{(1-p)^{y-1}}{p} \\
&= \frac{q}{1 - (1-p)(1-q)} \\
&= \frac{q}{p + q - pq}
\end{aligned}
$$

For the second, note that

$$
\begin{aligned}
P(X > Y) = 1 - P(Y \geq X) &= 1 - \frac{p}{p + q - pq} \\
&= \frac{q - pq}{p + q - pq}
\end{aligned}
$$

$\square$

Lemma 3. *Let $Y_1, \ldots, Y_t$ be independent Geometric random variables with respective parameters $q_1, \ldots, q_t$. Let $X \sim \mathrm{Geom}(p)$ be independent. Then*

$$P\left(X \geq \max_i Y_i\right) = \sum_{x=1}^{\infty} p(1-p)^{x-1} \prod_{i=1}^{t} \left(1 - (1-q_t)^x\right)$$

16

Proof.

$$P(X \geq \max_i Y_i) = \sum_{x=1}^{\infty} P(X = x \cap \max_i Y_i \leq x)$$

$$= \sum_{x=1}^{\infty} P(X = x) P(\max_i Y_i \leq x)$$

$$= \sum_{x=1}^{\infty} P(X = x) \prod_{i=1}^{t} P(Y_i \leq x)$$

by independence. Substitute the PMF and CMFs of the geometric distribution to finish. $\square$

Proof of Theorem 2.

Proof. Note that

$$A = \left\{ \frac{1}{B+1} \sum_{j=1}^{B+1} R'_j < \frac{1}{B} \sum_{j=1}^{B} R'_j \right\}$$

$$= \left\{ B \sum_{j=1}^{B+1} R'_j < (B+1) \sum_{j=1}^{B} R'_j \right\}$$

$$= \left\{ R'_{B+1} < \frac{1}{B} \sum_{j=1}^{B} R'_j \right\}$$

We now consider a subset of this event. We will compute the probability that $R'_1, \ldots, R'_B \in \mathcal{R}_b$ and that $R'_{B+1} \in \mathcal{R}_g$. By definition, this latter event falls into A.

For this to occur, note from Lemma 1 that each survival is drawn from a Geometric distribution. We want the chance that the B largest survivals are from set $\mathcal{R}_b$ and then the next largest is from $\mathcal{R}_g$. For additional simplicity, we further minimize the probability of this event by assigning all entries in $\mathcal{R}_g$ risk r_m and all entries in $\mathcal{R}_b$ risk $r_m + 3\varepsilon$.

Conditioning on $s'_{B+1} = x$, all entries from $\mathcal{R}_g$ are below x except for one, which is equal to x. There are $|\mathcal{R}_g|$ ways to pick this largest one.

Then $|\mathcal{R}| - B$ of the models in $\mathcal{R}_b$ are also below x, and B are above x. There are $\binom{|\mathcal{R}_b|}{B}$ ways to choose such B.

Finally this probability is summed over all values of x. This is

$$\sum_{x=1}^{\infty} |\mathcal{R}_g| (1 - (1 - r_m)^x)(1 - r_m)^{x-1} r_m \cdot \binom{|\mathcal{R}_b|}{B} (1 - (1 - r_m - 3\varepsilon)^x)^{|\mathcal{R}_b|-B} (1 - r_m - 3\varepsilon)^{Bx}$$

As before, to get a simpler expression, we just consider $x = 1$, yielding

$$P(A) \geq |\mathcal{R}_g| \binom{|\mathcal{R}_b|}{B} r_m^2 (r_m + 3\varepsilon)^{|\mathcal{R}_b|-B} (1 - r_m - 3\varepsilon)^B$$

$\square$

We also discuss the additional assumption of this Theorem to show that it is readily satisfied.

Assumption. There is a partition of $\mathcal{R} = \mathcal{R}_g \cup \mathcal{R}_b$ such that $\mathcal{R}_g = \{\mathbf{w} \in \mathcal{R} : r_m \leq R_{\mathcal{D}}(\mathbf{w}) \leq r_m + \varepsilon\}$ and $\mathcal{R}_b = \{\mathbf{w} \in \mathcal{R} : r_m + \varepsilon < R_{\mathcal{D}}(\mathbf{w}) \leq r_m + 3\varepsilon\}$, and that $|\mathcal{R}_b| \geq B$. That is, there is a set of *good* and *bad* models in terms of risk.

We can set $\varepsilon > 0$ to be any small number that satisfies a partition within the reservoir separating "good" from "bad" models – e.g. $\varepsilon = (\max_k R'_k - r_m)/4$ works. For demonstration, suppose the

17

underlying risks of the models in the reservoir range from 0.1 to 0.3, with 0.1 being the best risk r_m. Then $\varepsilon = 0.05$. $\mathcal{R}_g$ then contains the models with risks in $[0.1, 0.15]$ and $\mathcal{R}_b$ contains those with risks in $(0.15, 0.3]$. The only time this isn't satisfied for any ε is if one of the partitions always ends up empty, which only happens if all the K models have the same risk, a highly unlikely scenario where it is obvious that an ensemble isn't needed.

A.2   Learning bounds

Our bound follows from the following lemma from [22]. This itself is based on martingale inequalities from [41] and developments in [21].

Lemma 4 (Dekel). *Let $L_1, \ldots, L_T$ be a sequence of real-valued random variables and let $Z_1, \ldots, Z_T$ be a sequence of random variables such that $L_t = \mathbb{E}[L_t | Z_1, \ldots, Z_t]$. Also assume that $L_t \in [0, C]$ for all t. Define $U_t = \mathbb{E}[L_t | Z_1, \ldots, Z_{t-1}]$, and let $\tilde{L}_t = \sum_{i=1}^{t} L_i$. Then for any $T \geq 4$ and $\delta \in (0, 1)$,*

$$P\left(\forall t \in \{1, \ldots, T\}, \quad \sum_{i=1}^{t} U_i > \tilde{L}_t + \sqrt{2C \log\left(\frac{T}{\delta}\right)\tilde{L}_t} + 7C \log\left(\frac{T}{\delta}\right) \right) < \delta$$

From this we obtain our learning bound. The proof follows analogously to the one in [22], with modifications to handle our specific algorithm.

Proof of Theorem 3.

Proof. In our method we have a reservoir $\mathcal{R}$ containing K models $\mathbf{w}$ with associated weights $b_{\mathbf{w}}$. We focus on the case where the weight is equal to the passive steps survived by the model. To simplify this proof we consider a slightly modified algorithm, where all models with survival exceeding a threshold s are included in the reservoir.

$$\pi_t = \begin{cases} 0, & \text{if } s(\mathbf{w}_t) < s \\ s, & \text{if } s(\mathbf{w}_t) = s \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

We rewrite the weighted average of the reservoir models as a function of the entire training sequence of length T by defining π_t to be 1 if $\mathbf{w}_t$ is in $\mathcal{R}$ and 0 otherwise. Furthermore on the first instance that $\mathbf{w}_t$ joins $\mathcal{R}$, the π_t is instead set to s. This makes the weighting deterministic with respect to the information at time $t-1$, at the cost of the reservoir size not being explicitly defined. While this is analogous to a top-k averaging scheme, the full WRS scheme will follow from the same reasoning, with a more complicated definition of π_t. This is because all decision-making is performed using information that has already been seen.

Then we write $\mathbf{w}_{wrs} = \frac{1}{\sum_{t=1}^{T} \pi_t} \sum_{t=1}^{T} \pi_t \mathbf{w}_t$, and our goal is to control $R_{\mathcal{D}}(\mathbf{w}_{wrs})$. Furthermore, define the cumulative loss as $M_{wrs} = \sum_{t=1}^{T} \pi_t \ell(\mathbf{w}_{t-1}, z_t)$. (We note that this term is a little different from the observed cumulative WRS loss because of the requirement of reweighting on the s-th survival.)

In the lemma, let $L_t = \pi_t \ell(\mathbf{w}_{t-1}, z_t)$ and let $U_t = \mathbb{E}[\pi_t \ell(\mathbf{w}_{t-1}, z_t) | z_1^{t-1}]$. Noting that L_t is deterministic when conditioning on z_1^t, the assumptions are met. Then with probability $1 - \delta$, we have $\sum_{i=1}^{T} U_i < \tilde{L}_T + \sqrt{2C \log\left(\frac{T}{\delta}\right)\tilde{L}_T} + 7C \log\left(\frac{T}{\delta}\right)$.

Now $\sum_{i=1}^{T} U_i = \sum_{t=1}^{T} \pi_t \mathbb{E}[\ell(\mathbf{w}_{t-1}, z_t) | z_1^{t-1}] = \sum_{t=1}^{T} \pi_t R_{\mathcal{D}}(\mathbf{w}_{t-1})$, where the first equality uses the fact that π_t is determined by the filtration $\mathcal{F}_{t-1}$, and the second similarly uses the conditioning information to fix $\mathbf{w}_{t-1}$ and the i.i.d. assumption over z.

Finally, we have $R_{\mathcal{D}}(\mathbf{w}_{wrs}) = R_{\mathcal{D}}\left(\frac{1}{\sum_{t=1}^{T} \pi_t} \sum_{t=1}^{T} \pi_t \mathbf{w}_t \right) \leq \frac{1}{\sum_{t=1}^{T} \pi_t} \sum_{t=1}^{T} \pi_t R_{\mathcal{D}}(\mathbf{w}_t)$ from Jensen's inequality.

18

Dividing the previous statement by $\sum_{t=1}^{T} \pi_t$, we put the pieces together to conclude

$$R_{\mathcal{D}}(\mathbf{w}_{wrs}) \leq \frac{M_{wrs}}{\sum_t \pi_t} + \sqrt{\frac{2C \log\left(\frac{T}{\delta}\right) M_{wrs}}{(\sum_t \pi_t)^2}} + \frac{7C \log\left(\frac{T}{\delta}\right)}{\sum_t \pi_t}$$

Finally, note that $\sum_t \pi_t$ is the total time steps involved with the reservoir weights, which is at least $K_T s_T$ with K_T weights and s_T the minimal survival time of the reservoir. Substituting completes the proof.

$\square$

Proof of Theorem 4.

Proof. Combining the penultimate form of Theorem 3 (with $\sum_t \pi_t$ instead of $K_T s_T$) and the deduction that $M_{wrs}/\sum_t \pi_t \leq M_T/T$ we have

$$R_{\mathcal{D}}(\mathbf{w}_{wrs}) \leq \frac{M_T}{T} + \sqrt{\frac{2C \log\left(\frac{T}{\delta}\right) M_T}{T \sum_t \pi_t}} + \frac{7C \log\left(\frac{T}{\delta}\right)}{\sum_t \pi_t}$$

Next, we turn our attention to the generic base-model online regret bound. From this we get $M_T/T \leq \frac{1}{T} \sum_t \ell(\mathbf{w}^*; z_t) + r(T)/T$.

We need to bound the cumulative loss of $\mathbf{w}^*$, that is $\frac{1}{T} \sum_t \ell(\mathbf{w}^*; z_t)$, in terms of the risk: $R_{\mathcal{D}}(\mathbf{w}^*)$. Using an application of Hoeffding bound, we get

$$\frac{1}{T} \sum_{t=1}^{T} \ell(\mathbf{w}^*, z_t) \leq R_{\mathcal{D}}(\mathbf{w}^*) + C \sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2T}}$$

which holds with probability $1 - \delta$.

Combining the undesired tail events under the union bound, we get

$$R_{\mathcal{D}}(\mathbf{w}_{wrs}) \leq R_{\mathcal{D}}(\mathbf{w}^*) + \frac{r(T)}{T} + \sqrt{\frac{2C \log\left(\frac{T}{\delta}\right) M_T}{T \sum_t \pi_t}} + \frac{7C \log\left(\frac{T}{\delta}\right)}{\sum_t \pi_t} + C \sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2T}}$$

with probability $1 - 2\delta$. As before, substitute in $K_T s_T$ for $\sum_t \pi_t$ to complete.

$\square$

Remark. The use of the online regret bound also indicates that the base models $\mathbf{w}_t$ are improving over time. This means that their risks are decreasing, and hence the observed survival times are increasing. Thus even in the error term without T in the denominator, s_T is an increasing function of T so we expect the error terms to decay in T.

19

B Additional details on methods and experimental setups

B.1 Datasets

Of the 16 datasets used in our study (see Table 1), 14 are directly from existing repositories and/or literature, while 2 of them — Newsgroups (Binary, CS) and SST-2 — were modified from pre-existing assets. First, Newsgroups (binary, CS) was formed by taking the original multi-class Newsgroups dataset hosted in [31] and combining the labels "Computers" and "Science" as a $+1$ class, and all other labels as the -1 class. Second, SST-2 is a sentiment classification dataset originally containing text excerpts that we transformed into a linear classification dataset by using the `CountVectorizer` from [31].

B.2 Base PAC and FSOL hyperparameter tuning

We start our set of experiments by tuning the C_{err} hyperparameter for base PAC, testing values in the set $10^{-3}, 10^{-2}, \ldots, 10^2, 10^3$. For base FSOL, following Zhao et al.'s approach, we test η values in $2^{-3}, 2^{-2}, \ldots, 2^8, 2^9$ and λ values in $0, 10^{-3}, 10^{-2}, \ldots, 10^2, 10^3$. For each dataset, we perform five runs of our base PAC and FSOL variants with randomized shuffles of the training and test data, running through the training data only once for each run. For each dataset, we pick the C_{err}, η, and λ values corresponding to the base PAC and FSOL variants, within the top 2.5% in terms of final test accuracy, that had the highest ROP, with all metrics averaged across the five runs. This experimental choice was made to strike a balance between simulating as difficult and risky conditions as possible and still choosing useful base model variants.

B.3 Metrics logging

Given that some of our datasets contain up to 36 million training points, it would be computationally and storage-wise unfeasible to record metrics at each timestep. As such, for each dataset, we record metrics at ≈ 200 evenly-spaced timesteps (exact number depends on divisibility and integer division of N_{train} by 200) throughout the training stream, in addition to the initial and final timesteps. As such ROP was also approximated by proxying oracle accuracy by taking the cumulative maximum base model test accuracy across timesteps where metrics were recorded. As shown in the over-time figures in Appendix G.2, such a reduced resolution still tells us a very clear picture, while cutting down computation and storage requirements by many orders of magnitude.

B.4 Compute requirements and code availability

All experiments were run on a Linux computing cluster with 32 nodes, each with 40 Intel Xeon E5-2650 CPU cores and a total of 500 GB of RAM per node, managed using SLURM. Nonetheless, no experiments require multiprocessing or multiple cores. Depending on dataset size, some trials could take less than a minute to run, while the largest datasets would take a couple hours at max. However, larger datasets like Criteo will require 32 GB of RAM to load the dataset into memory.

Our source code for reproducing all experiments can be found at https://github.com/FutureComputing4AI/Weighted-Reservoir-Sampling-Augmented-Training/tree/main. All experiments were run on CPU.

20

C   Applications to Malware Classification

Given that the original motivation for this work was a real-world need in malware detection, we present empirical results of PAC, PAC-WRS, FSOL, and FSOL-WRS on the EMBER malware classification dataset [7]. The EMBER dataset is comprised of features extracted from 1.1M benign and malicious Windows portable executable files and thus can be considered very close to a real-world test case of our WRS-Augmented Training method on a real-world deployment setting.

In Figure 5, we observe that both base PAC and FSOL experience significant and frequent test accuracy fluctuations throughout the training process. However, both PAC-WRS and FSOL-WRS very successfully mitigate these test accuracy fluctuations and even outperform the oracle model, all without the use of a separate validation set.

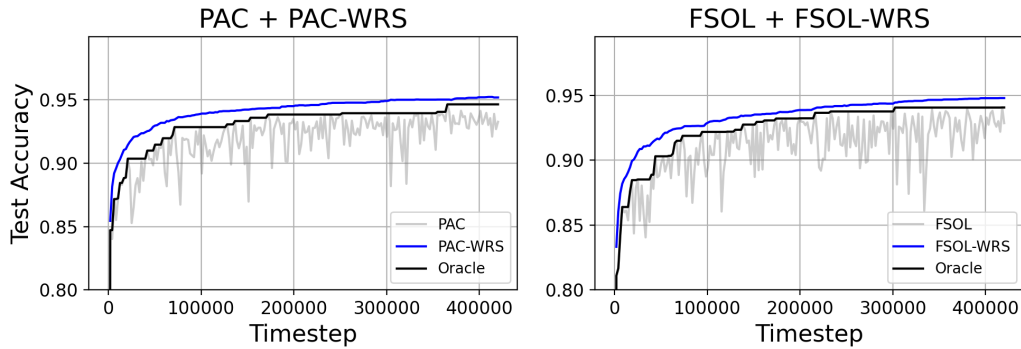



Figure 5: Test accuracies (y-axis) over timestep (x-axis) for PAC/FSOL and PAC/FSOL-WRS on the EMBER benchmark dataset for malware classification.

Indeed, WRS-Augmented Training is a promising tool towards stabilizing test accuracy performance in real-world deployment settings like malware detection.

21

D Comparisons to other ensembling and averaging schemes

One natural question a reader may ask is: how does WRS-Augmented Training compare to traditional weight-averaging schemes such as moving average (e.g., averaging the most-recently-observed $K = 64$ weight vectors at each timestep) or exponential average (e.g., forming an ensemble vector $\bar{\mathbf{w}}_t = \gamma \mathbf{w}_t + (1 - \gamma)\bar{\mathbf{w}}_{t-1}$ at each timestep), where $\mathbf{w}_t$ is the base algorithm's candidate solution at timestep t.

From Figures 6 and 7, we observe that the exponential average scheme is consistently ineffective at mitigating the test accuracy instabilities of the base models PAC and FSOL. The stability of the exponential average scheme is usually not much better than that of the base model, which makes sense because it still puts the majority of the weighting on the most recent candidate solution. From these figures, we also observe that the $K = 64$ moving average also has very mixed effectiveness. Overall, we see that WRS-Augmented Training is more preferable to the $K = 64$ moving average. This makes sense because with WRS-Augmented Training, we are much more selective about the quality of the candidate solutions that we retain in our reservoir, compared to moving average, which necessarily by definition must include poor-performing solutions as they appear.

Third, from Table 5, we see that for datasets with dimension $D > 100K$, the moving average method can be significantly computationally slower per iteration than WRS-Augmented Training. For example, with PAC as the base model, the moving average method was, on average, 6.579x slower per iteration than WRS-Augmented Training on KDD2010 (Algebra) and 10.050x slower per iteration than WRS-Augmented Training on URL. Similar trends hold when using FSOL as the base algorithm. These results make sense because with WRS-Augmented Training, we do not always add candidate solutions to our reservoir for averaging, while with the moving average method, we must always add new candidate solutions into our set. These insertion and deletion costs will accrue over time. On smaller datasets, the moving average is faster or slower depending on the dataset, but runtime is dominated by IO and all methods finish within minutes. However, in more real-world, large-scale settings where evaluation and checking are prohibitively expensive, WRS-Augmented Training is the fastest, most accurate, and most reliable method compared to all the baselines.

22

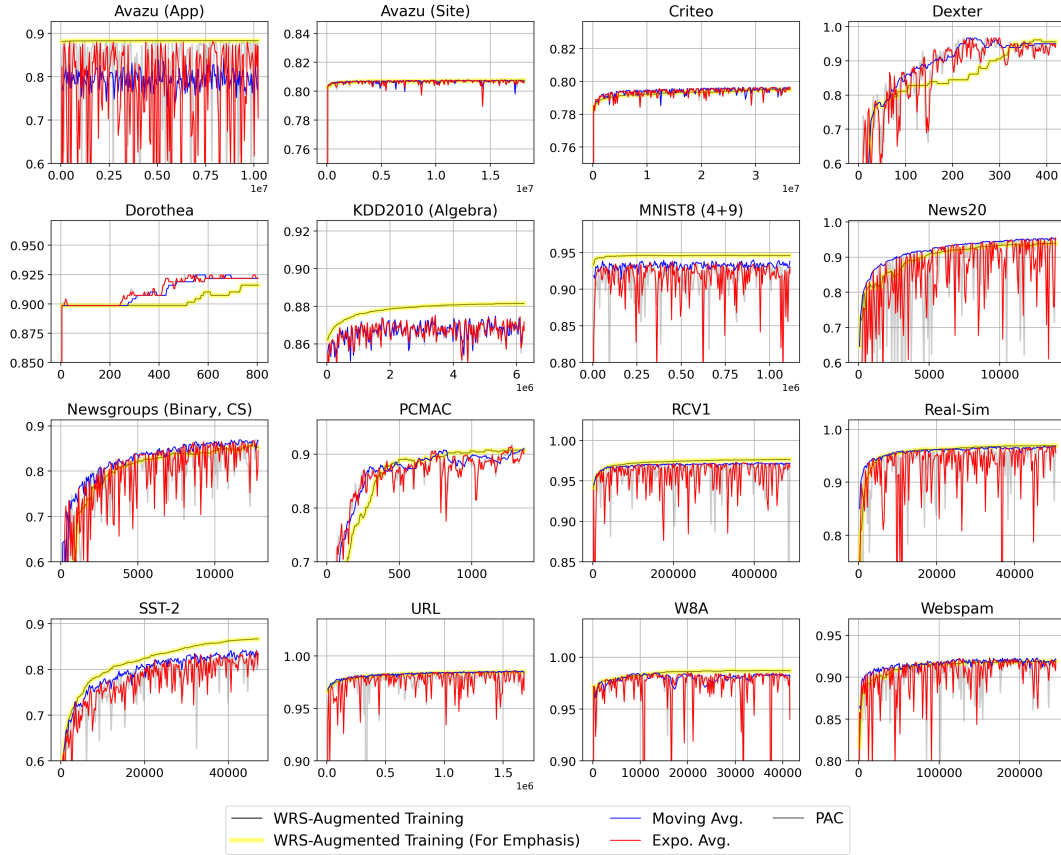



Figure 6: Test accuracies (y-axis) over timestep (x-axis) for WRS-Augmented Training ($K = 64$), moving average (most recent $K = 64$) and exponential average ($\gamma = 0.9$), using PAC as the base.

23

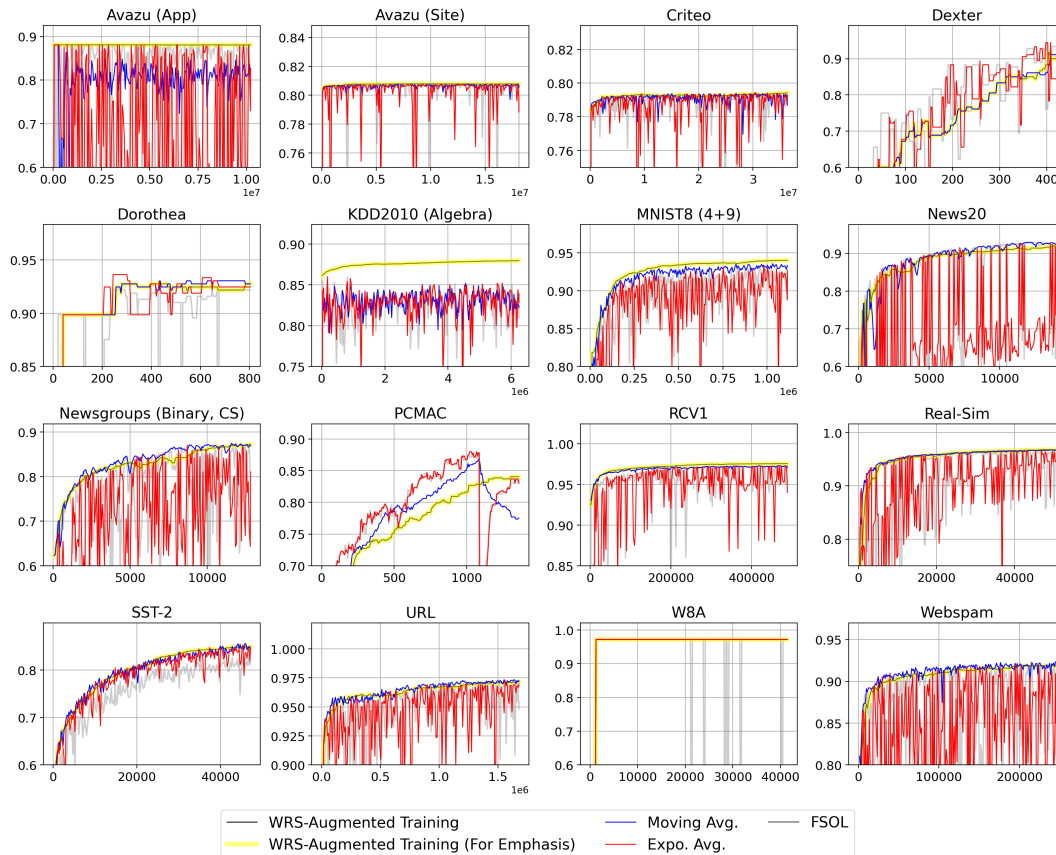



Figure 7: Test accuracies (y-axis) over timestep (x-axis) for WRS-Augmented Training ($K = 64$), moving average (most recent $K = 64$) and exponential average ($\gamma = 0.9$), using FSOL as the base.

Table 5: Compute times per iteration (in seconds) of moving average and exponential average **relative to WRS-augmented training**, with PAC (left) and FSOL (right) as base models. For example, on URL with PAC, *moving average was, on average,* $10.050x$ *slower per iteration than WRS-augmented training.* Results were averaged over 5 randomly-seeded trials.

		PAC		FSOL	
Dataset	**D**	**Moving Avg.**	**Expo. Avg.**	**Moving Avg.**	**Expo. Avg.**
Webspam	254	0.953	0.717	0.738	0.613
W8A	300	0.420	0.353	0.368	0.336
MNIST8 (4+9)	784	1.125	0.885	0.968	0.887
PCMAC	3289	0.483	0.312	0.377	0.235
SST-2	13757	0.903	0.431	0.748	0.411
Dexter	20000	0.577	0.143	0.285	0.066
Real-Sim	20958	0.949	0.624	0.707	0.564
RCV1	47236	1.825	0.892	1.366	1.011
Dorothea	100000	1.194	0.190000	0.580	0.254
Newsgroups (Binary, CS)	101631	3.587	1.042	1.376	0.543
Avazu (App)	1000000	8.357	2.507	8.078	2.544
Avazu (Site)	1000000	18.704	3.207	11.612	3.683
Criteo	1000000	13.108	2.876	7.009	2.381
News20	1355191	4.288	1.356	3.376	1.153
URL	3231961	10.050	3.945	9.124	4.058
KDD2010 (Algebra)	20216830	6.579	2.402	6.778	2.301

24

E   Modified WRS-Augmented Training on non-passive aggressive online learning methods

In this section, we explore applying a modified form of WRS-Augmented Training (WAT) on top of three non-passive-aggressive online learning methods: Stochastic Gradient Descent with Momentum [40], ADAGRAD [14], and Truncated Gradient Descent [15], all using default hyperparameters. We emphasize that these three methods are not passive-aggressive as they always update their weight vectors at each time step, even if the data point was correctly classified. As such, the original WAT is not directly applicable.

However, we can modify WAT as follows. First, we define a *pseudo-passive* step as one where the current solution candidate made no classification error (i.e., no more concept of margin). Second, at a time step when the current solution candidate does make an error, we will sample the last weight vector before a mistake was made into our reservoir with probability proportional to the number of pseudo-passive steps, before resetting our counter. The rest of WAT operates as normal under this pseudo-passive step weighting.

From Figures 8 - 10, we observe that all three non-passive-aggressive online algorithms, in particular Stochastic Gradient Descent with Momentum (SGD+M) and Truncated Gradient Descent (TGD), are susceptible to experiencing concerning fluctuations in test accuracy. Consistently, across the algorithms, we observe that when such fluctuations are present, modified WAT effectively mitigates such fluctuations very well. On the other hand, when there is little fluctuation in the base model (e.g., see ADAGRAD), applying modified WAT will do little to no harm. We emphasize that WAT was not designed for non-passive-aggressive algorithms, but is still demonstrably useful and adaptable to a wider class of base models, which can constitute fruitful future work.

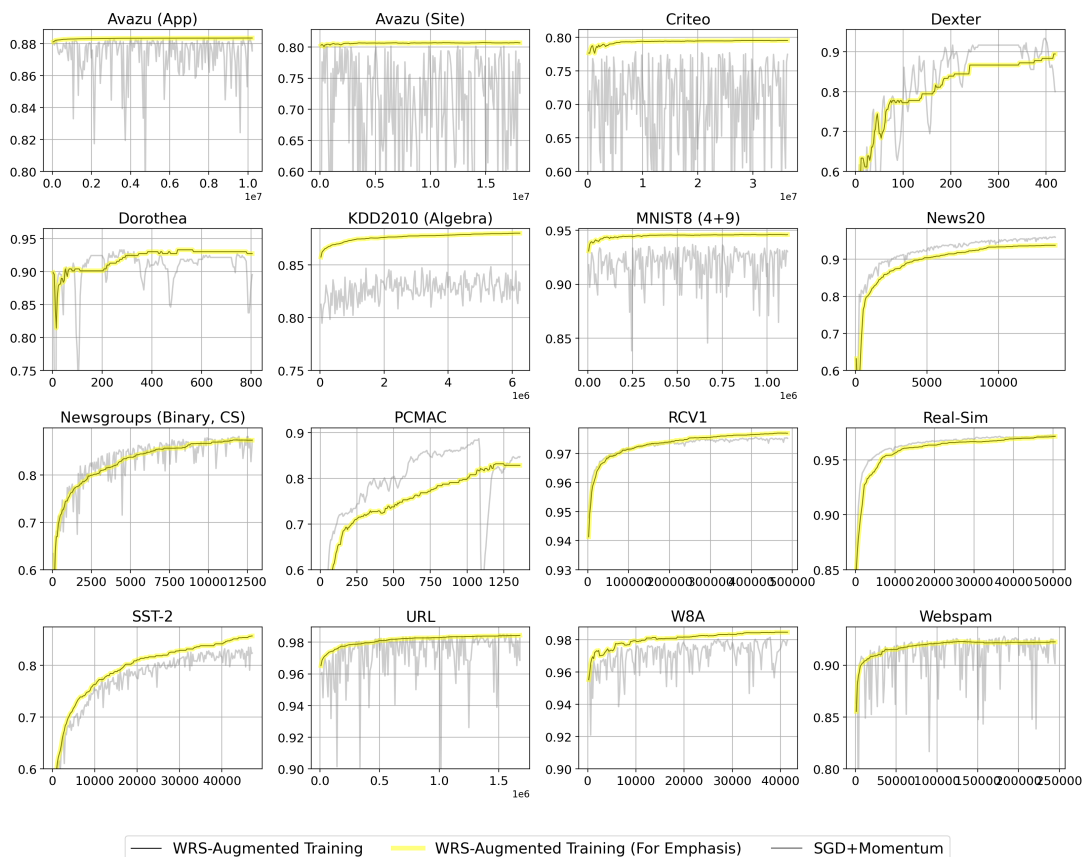



Figure 8: Test accuracies (y-axis) over timestep (x-axis) for modified WRS-Augmented Training ($K = 64$) on Stochastic Gradient Descent with Momentum.

25

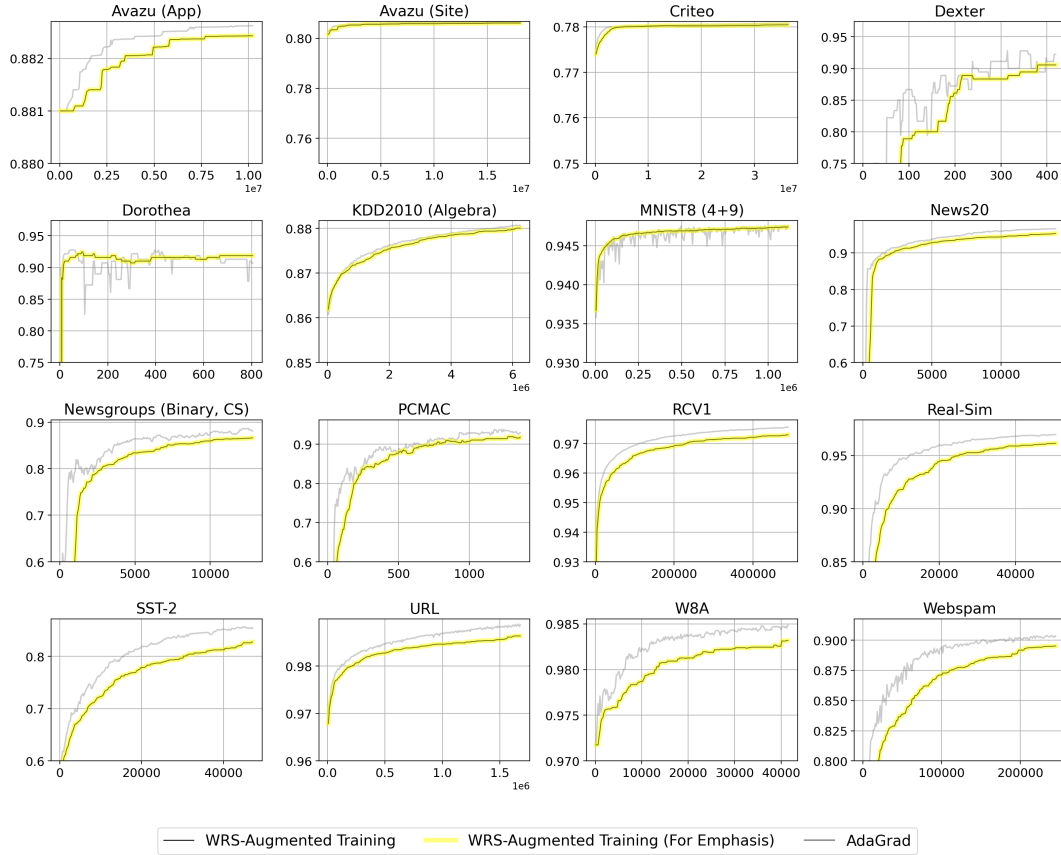



Figure 9: Test accuracies (y-axis) over timestep (x-axis) for modified WRS-Augmented Training ($K = 64$) on ADAGRAD.

26

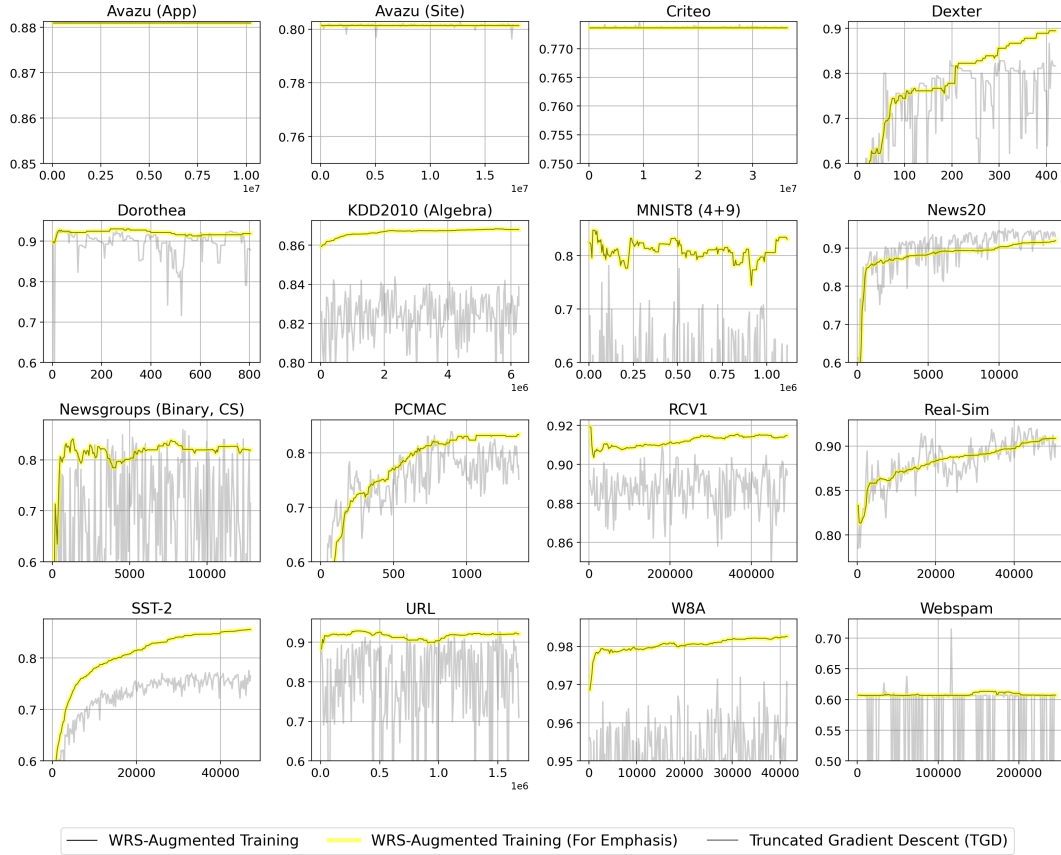



Figure 10: Test accuracies (y-axis) over timestep (x-axis) for modified WRS-Augmented Training ($K = 64$) on Truncated Gradient Descent.

27

F   Tables of p-values for Wilcoxon Signed-Rank Tests

Below, we provide full tables of p-values computed using Wilcoxon Signed-Rank Tests towards probing the statistical significance of the differences in ROP and final test accuracy between PAC/FSOL-WRS and their base model counterparts.

Table 6:   Wilcoxon signed-rank test p-values testing whether differences in **relative oracle performance** between $K = 64$ PAC/FSOL-WRS variants and base PAC/FSOL methods are statistically significant.

	Simple Average		Weighted Average	
	Standard	Exponential	Standard	Exponential
PAC	0.0150	0.0199	0.0386	0.325
FSOL	0.000437	0.0879	0.000437	0.0787

Table 7:   Wilcoxon signed-rank test p-values testing whether differences in **final test accuracy** between $K = 64$ PAC/FSOL-WRS variants and base PAC/FSOL methods are statistically significant.

	Simple Average		Weighted Average	
	Standard	Exponential	Standard	Exponential
PAC	0.408	0.196	0.501	0.856
FSOL	0.00836	0.313	0.00836	0.679

28

G Additional results figures for PAC-WRS and FSOL-WRS

G.1 Uncertainty-quantified aggregate metrics across runs

Below, we provide figures with error bars (minimum and maximum across 5 trials) showing the relative oracle performances (ROP), final test accuracies, and final sparsities of PAC-WRS and FSOL-WRS alongside the base PAC and FSOL models, across all 16 datasets. The main takeaways are that a) PAC-WRS and FSOL-WRS overall incur substantially-lower ROP than their corresponding base models; b) PAC-WRS and FSOL-WRS overall achieve comparable, if not improved, final test accuracy compared to their corresponding base models, consistently with lower variance, too; c) the final sparsities of PAC-WRS and FSOL-WRS are overall comparable, if not higher than those of their corresponding base models.

G.1.1 FSOL and FSOL-WRS

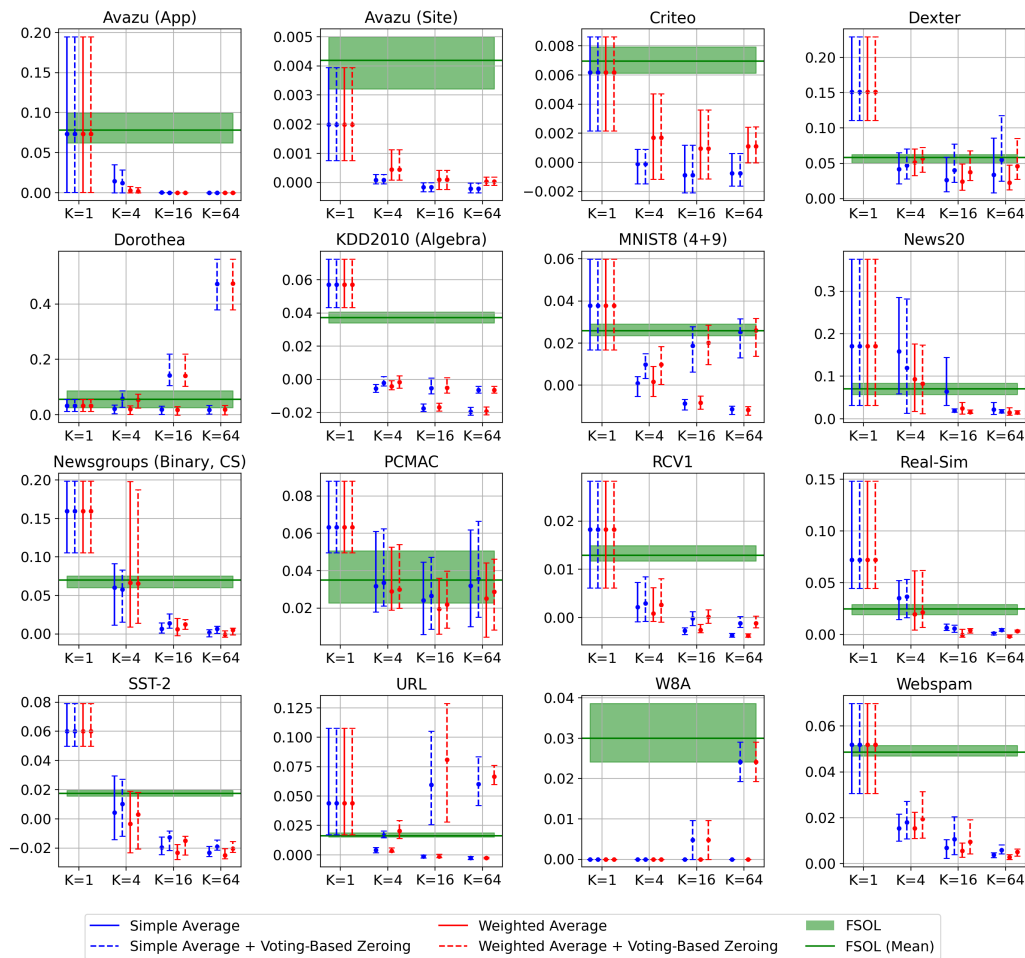



Figure 11: Relative oracle performances (y-axis) of base FSOL and FSOL-WRS using **standard** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity. **Lower values indicate more stable performance.**

29

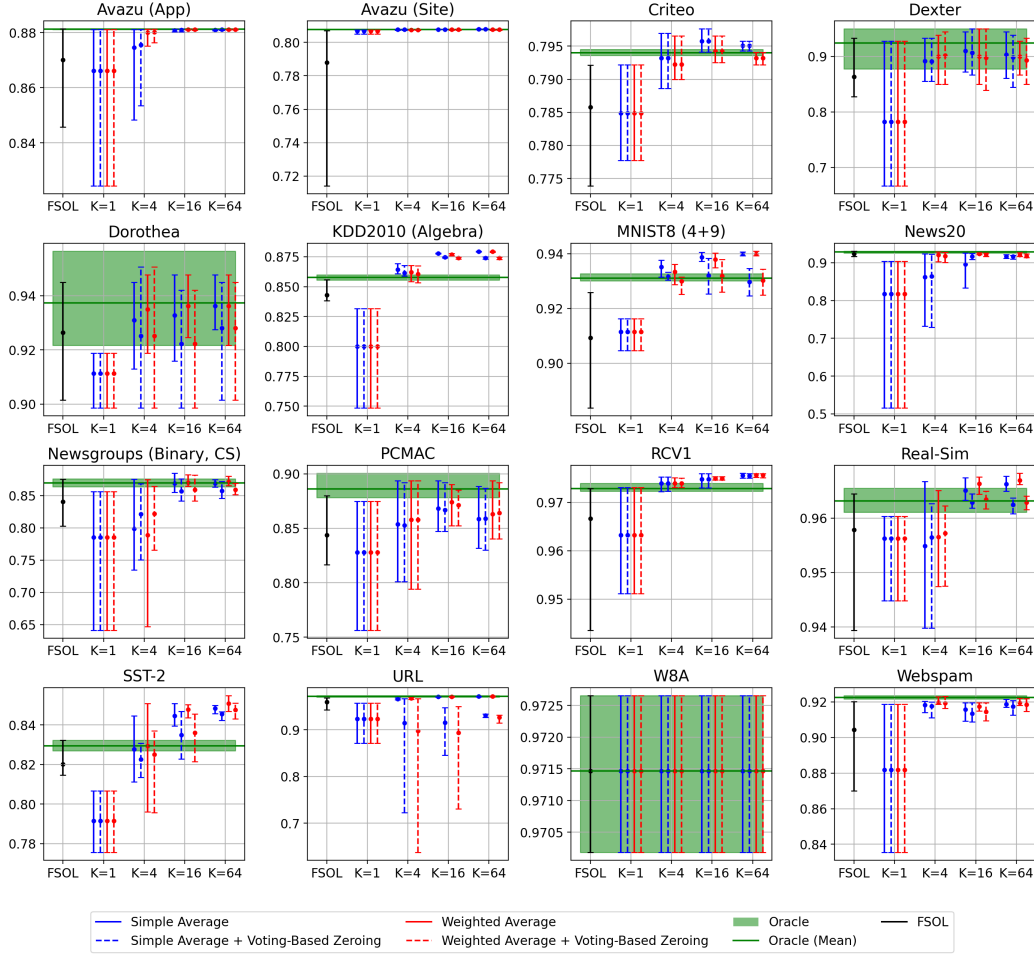



Figure 12: Final test accuracies (y-axis) of base FSOL and FSOL-WRS using **standard** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity.

30

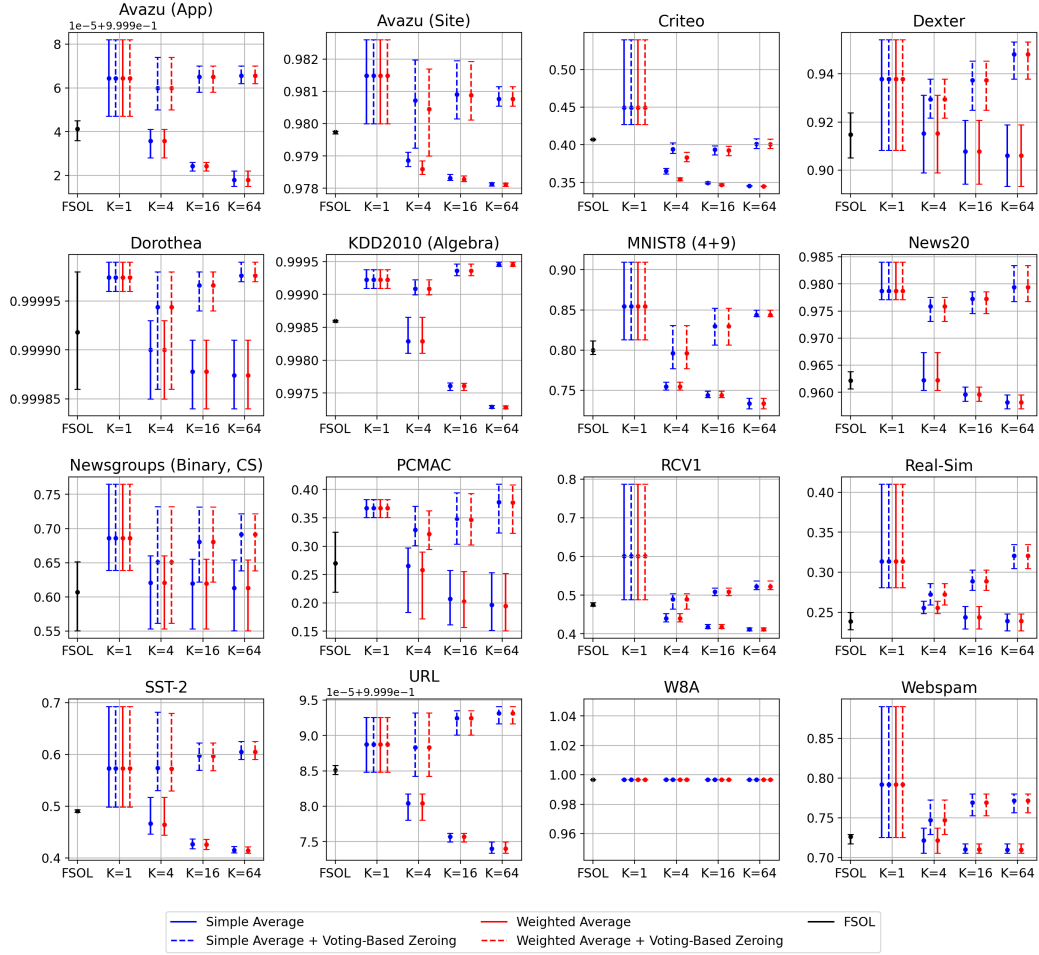



Figure 13: Final sparsities (y-axis) of base FSOL and FSOL-WRS using **standard** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity.

31

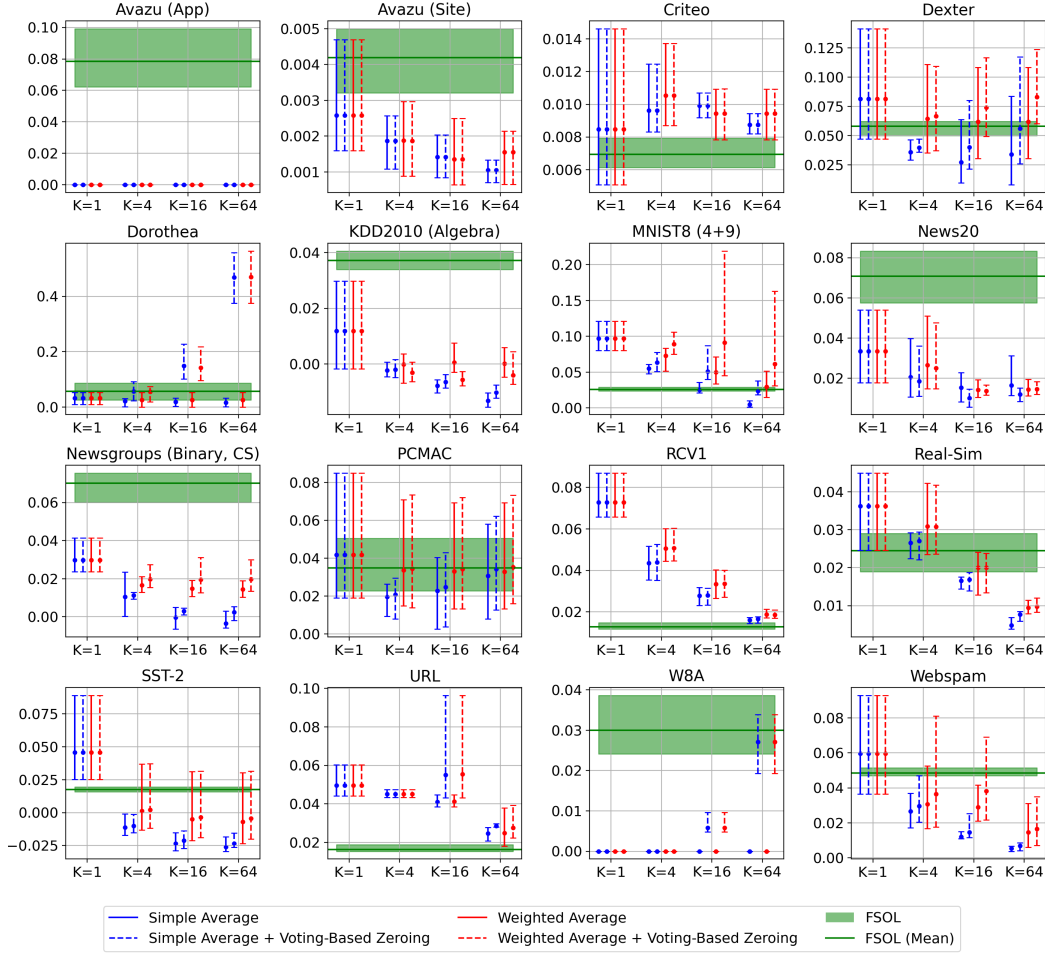



Figure 14: Relative oracle performances (y-axis) of base FSOL and FSOL-WRS using **exponential** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity. **Lower values indicate more stable performance.**

32

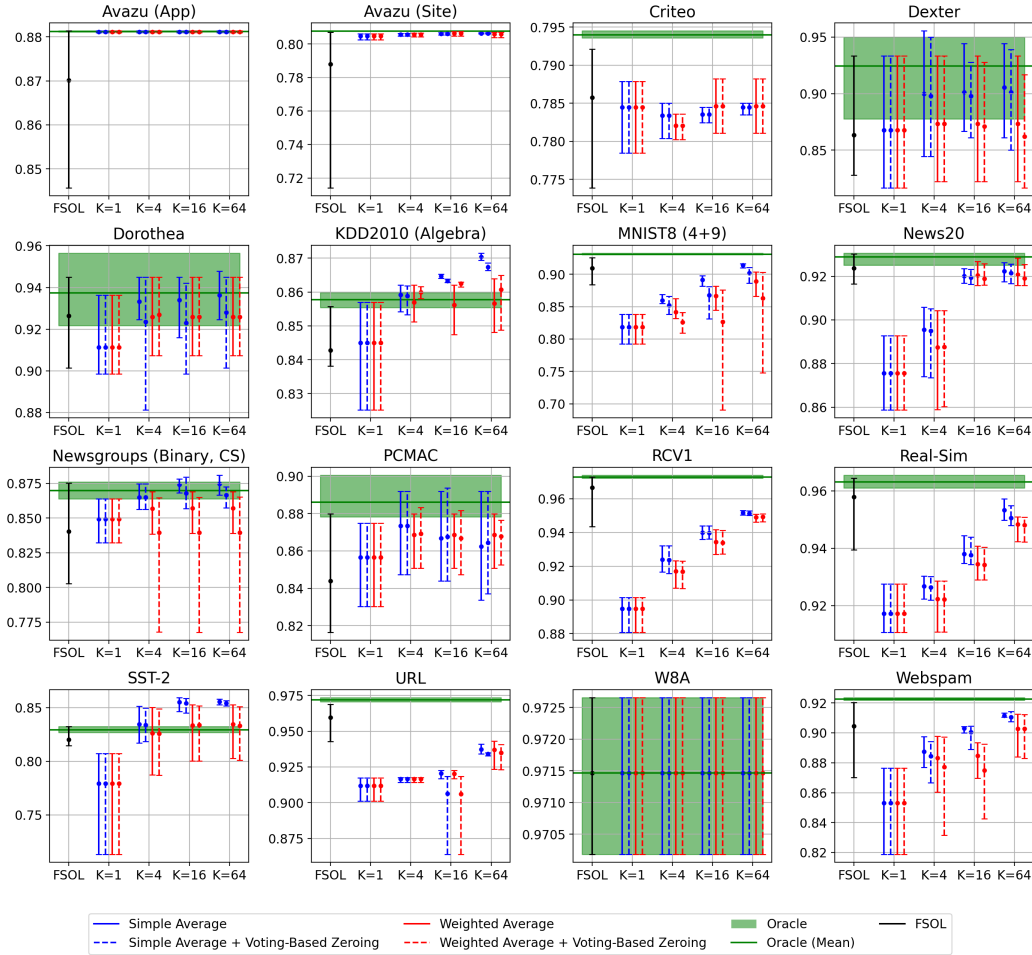



Figure 15: Final test accuracies (y-axis) of base FSOL and FSOL-WRS using **exponential** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity.

33

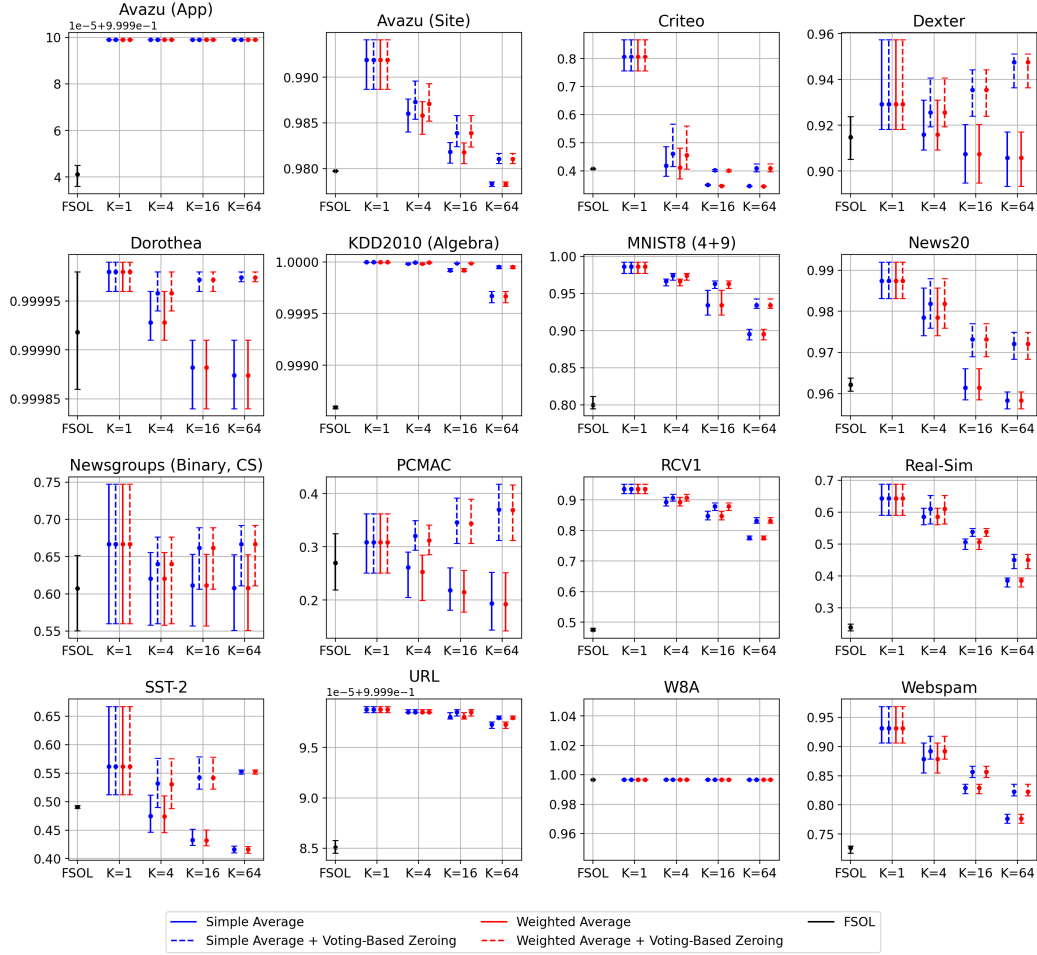



Figure 16: Final sparsities (y-axis) of base FSOL and FSOL-WRS using **exponential** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity.

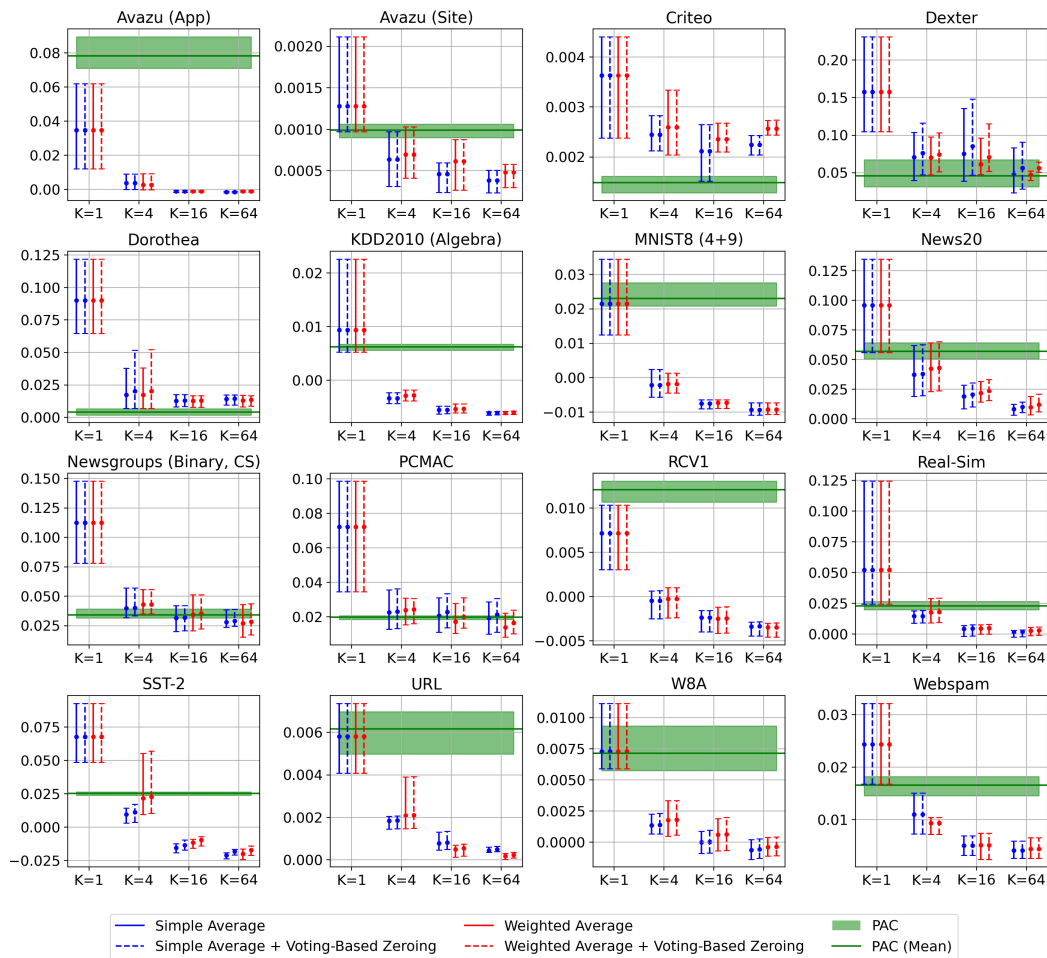
34

G.1.2 PAC and PAC-WRS

Figure 17: Relative oracle performances (y-axis) of base PAC and PAC-WRS using **standard** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity. **Lower values indicate more stable performance.**

35

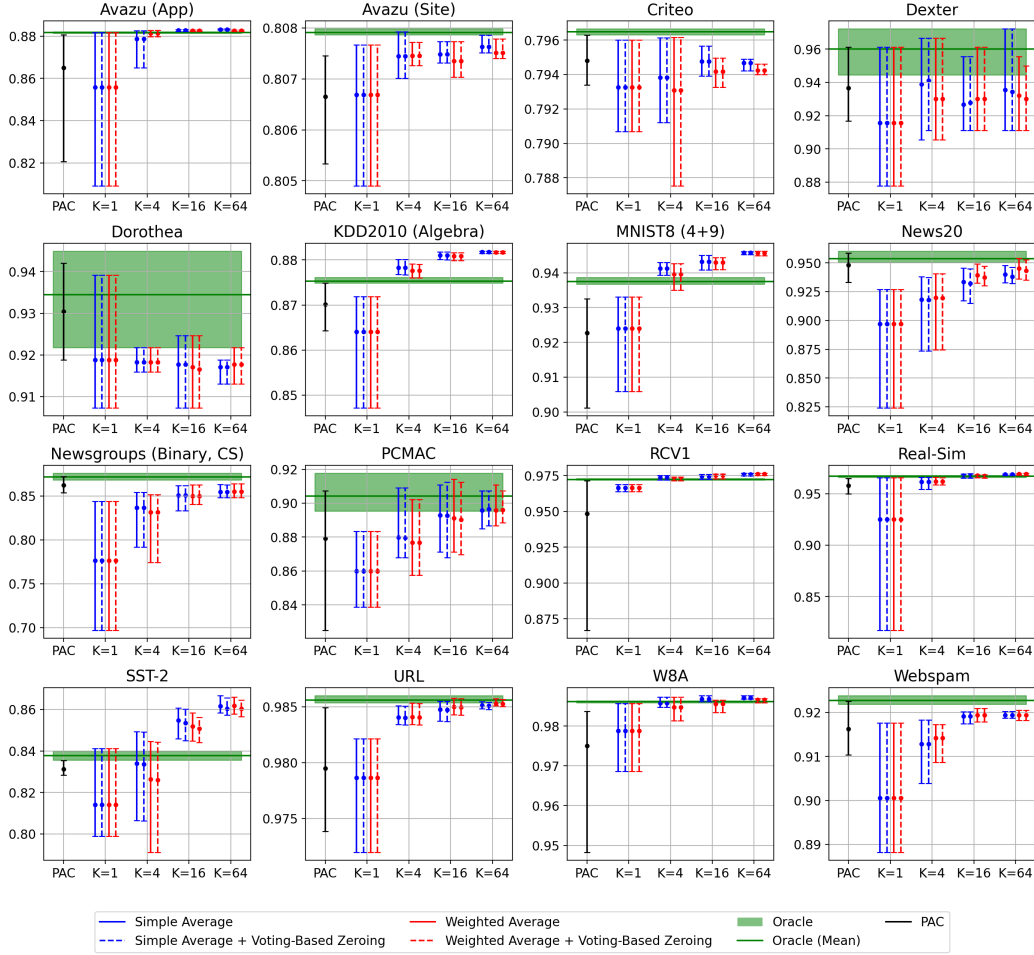



Figure 18: Final test accuracies (y-axis) of base PAC and PAC-WRS using **standard** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity.

36

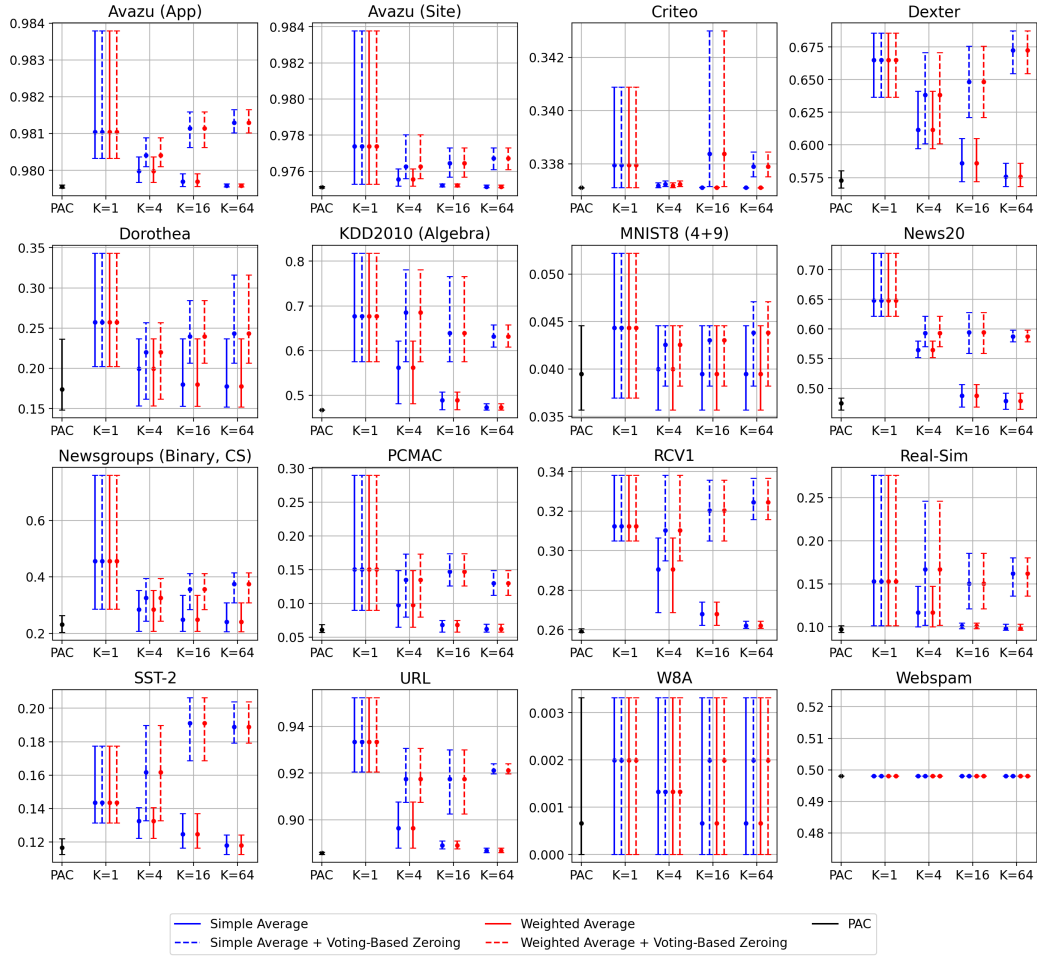



Figure 19: Final sparsities (y-axis) of base PAC and PAC-WRS using **standard** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity.

37

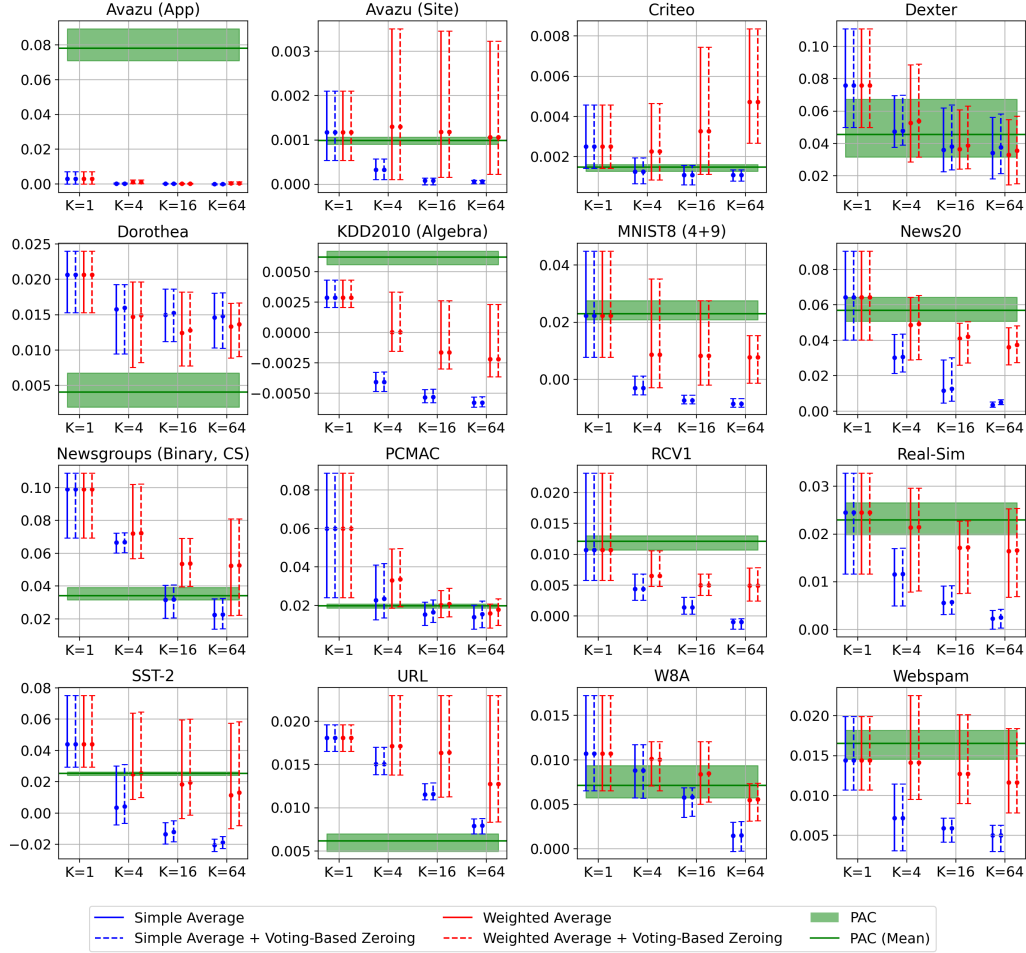



Figure 20: Relative oracle performances (y-axis) of base PAC and PAC-WRS using **exponential** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity. **Lower values indicate more stable performance.**

38

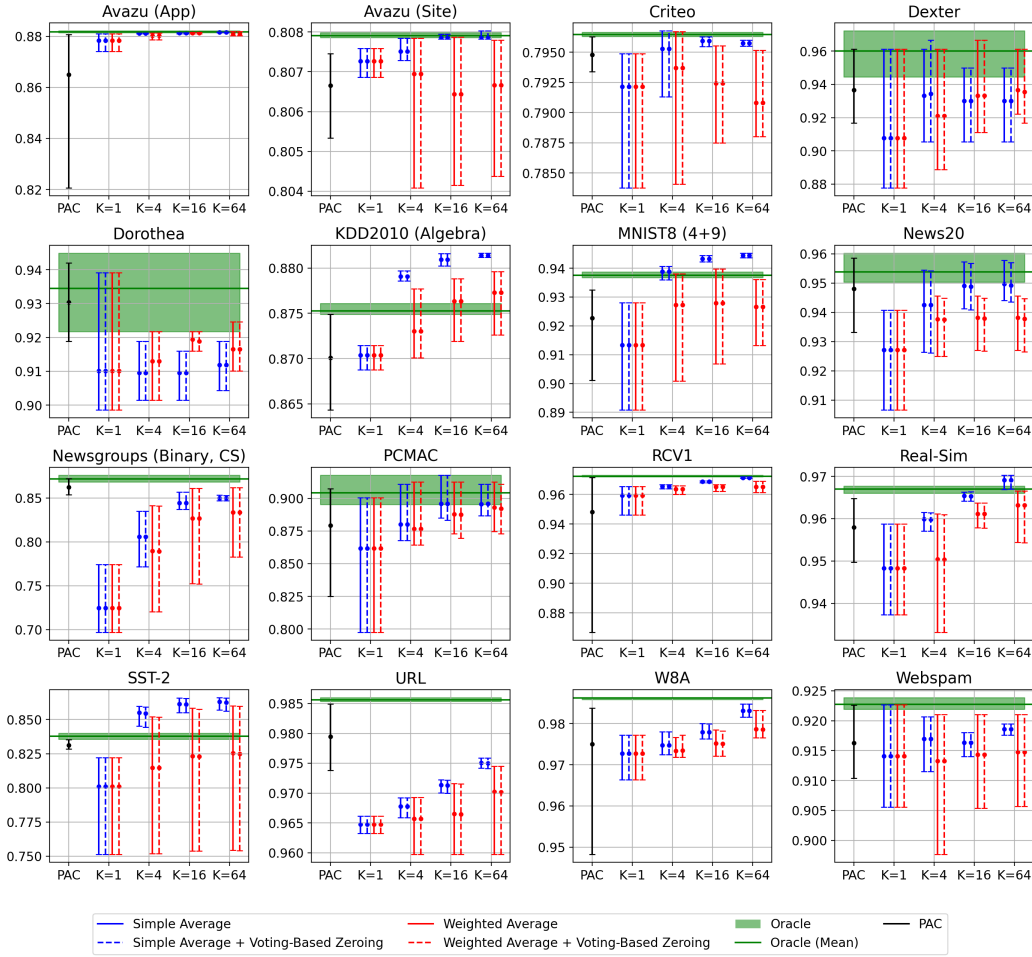



Figure 21: Final test accuracies (y-axis) of base PAC and PAC-WRS using **exponential** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity.

39

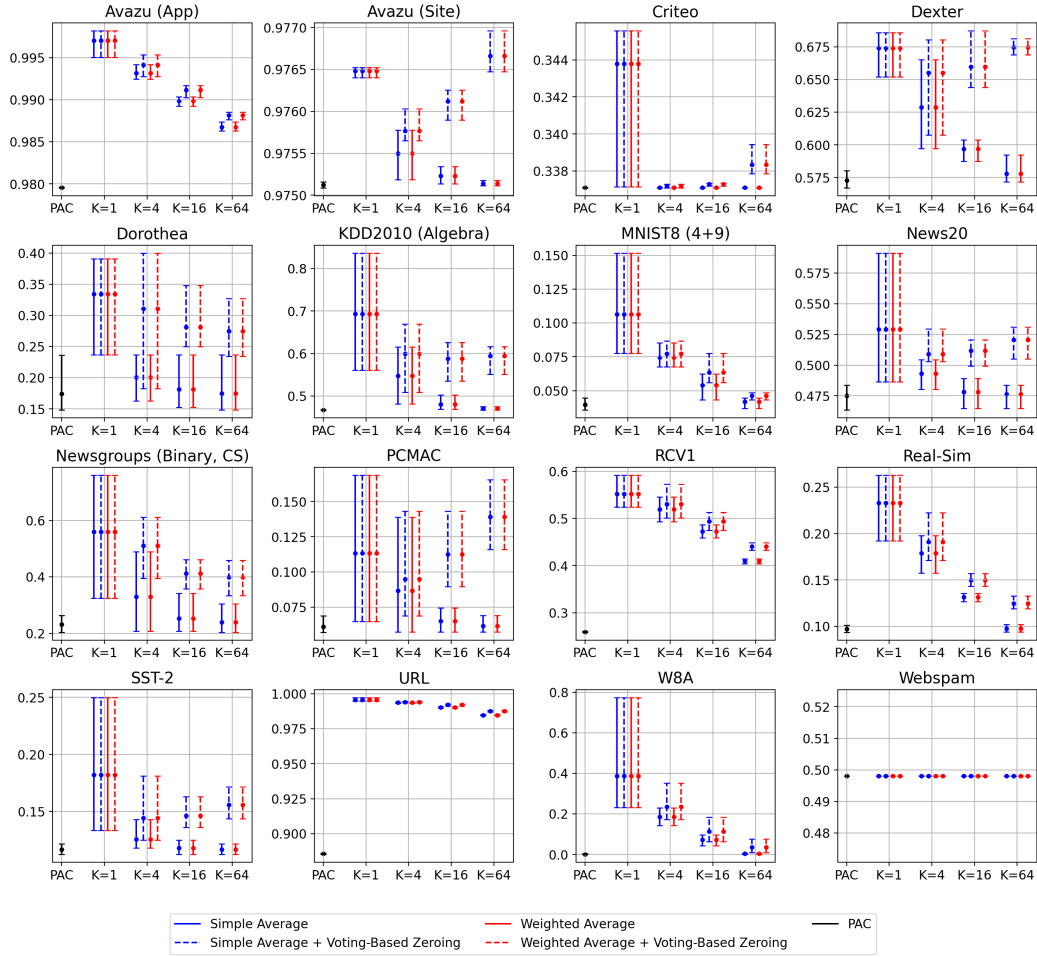



Figure 22: Final sparsities (y-axis) of base PAC and PAC-WRS using **exponential** weights over reservoir sizes K (x-axis) on all datasets. Error bars represent the minimum and maximum values achieved across 5 randomized trials. **Blue:** WRS-augmented variants via simple average of reservoir members. **Red:** WRS-augmented variants via weighted average of reservoir members. **Dotted lines:** indicates voting-based zeroing was performed for additional sparsity.

40

G.2 Test accuracies and sparsities over time

Below, we provide figures showing test accuracies and sparsities over time on individual runs of PAC-WRS and FSOL-WRS, compared against their base model counterparts, on all 16 datasets. The main takeaways are that a) test accuracies over time on PAC-WRS and FSOL-WRS are overall much stabler, if not also higher, than their base model counterparts (and much closer to that of the oracle); b) applying WAT to PAC and FSOL does not significantly impact sparsity — in some cases, especially with voting-based zeroing, sparsity actually increases compared to the base model!

G.2.1 FSOL and FSOL-WRS

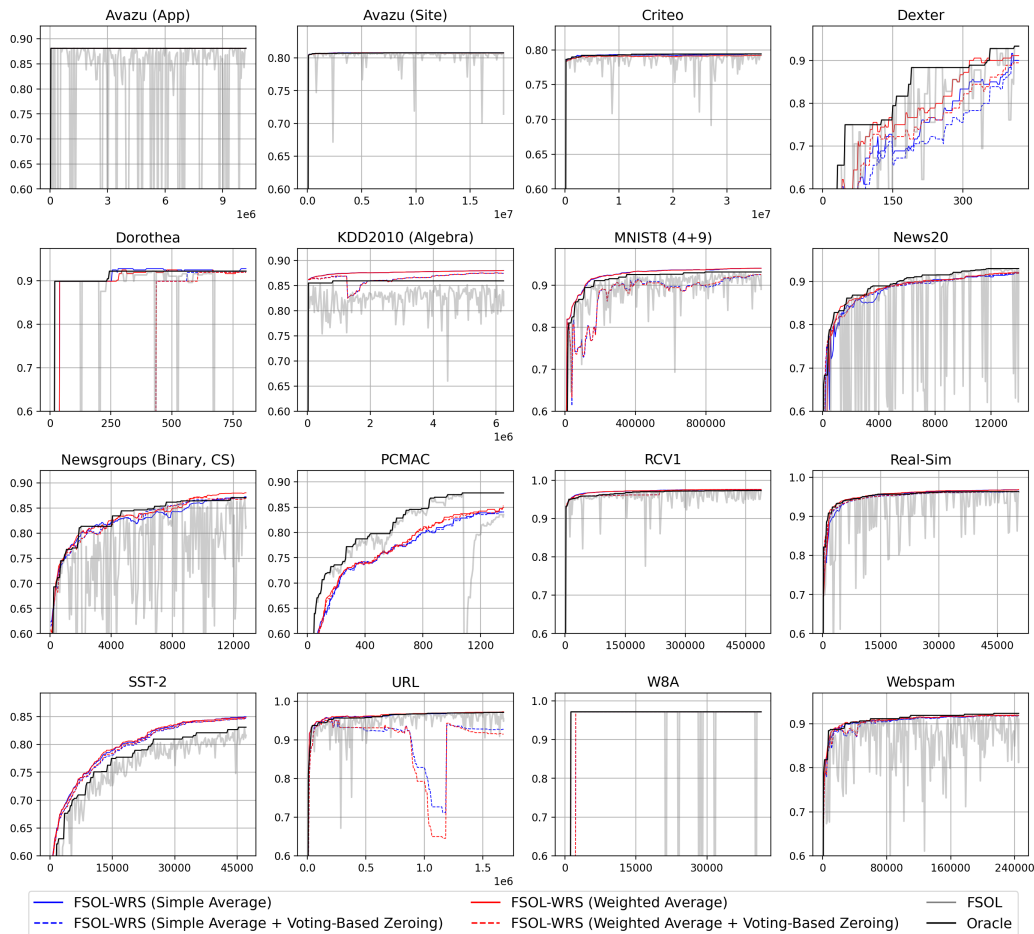



Figure 23: Test accuracies (y-axis) over timestep (x-axis) for FSOL-WRS with reservoir size $K = 64$ using standard weighting on all 16 tested datasets. **Light grey lines:** test accuracies of the FSOL baseline methods at each timestep. **Solid black lines:** test accuracies of the "oracle" models, computed as the cumulative maximum of the FSOL baselines. **Blue:** corresponds to FSOL-WRS variants ensembled via simple averaging. **Red:** corresponds to FSOL-WRS variants ensembled via weighted averaging. **Dotted lines:** indicate whether voting-based zeroing was applied for additional sparsity.

41

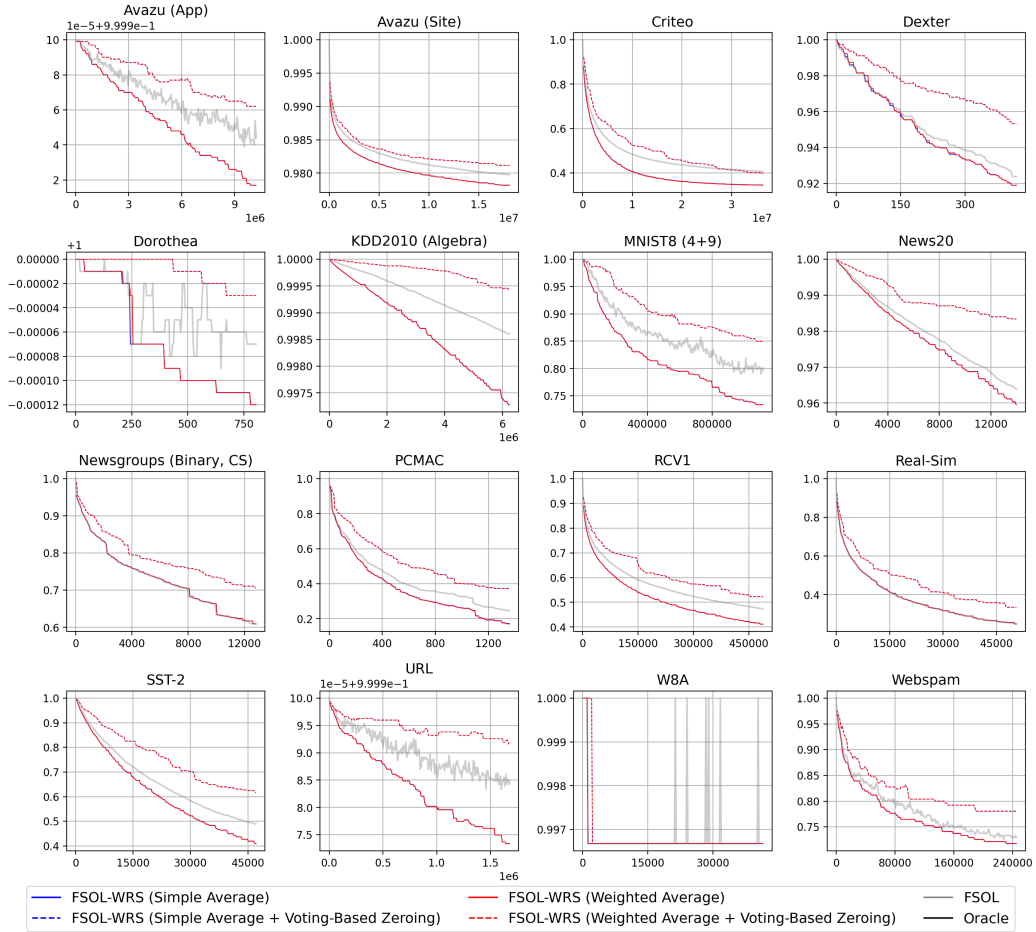



Figure 24: Sparsity (y-axis) over timestep (x-axis) for FSOL-WRS with reservoir size $K = 64$ using standard weighting on all 16 tested datasets. **Light grey lines:** sparsities of the FSOL baseline methods at each timestep. **Blue:** corresponds to FSOL-WRS variants ensembled via simple averaging. **Red:** corresponds to FSOL-WRS variants ensembled via weighted averaging. **Dotted lines:** indicate whether voting-based zeroing was applied for additional sparsity.

42

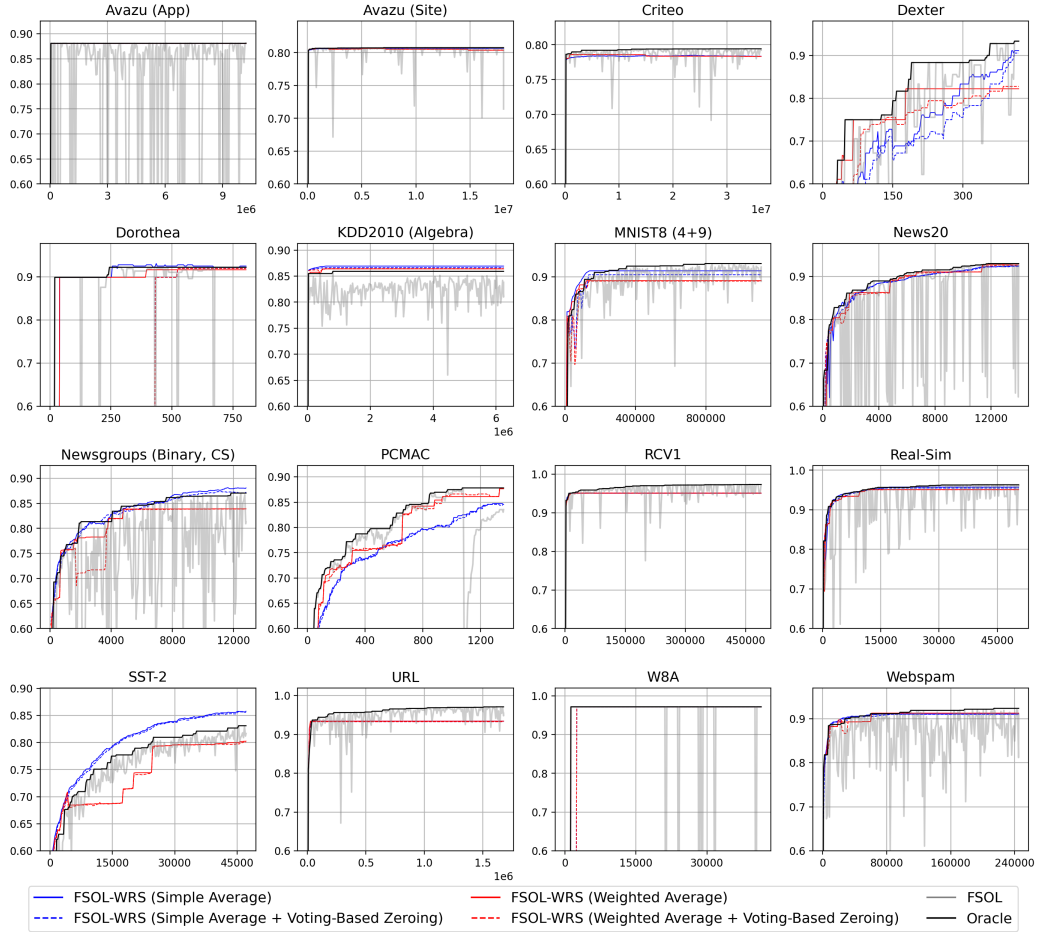



Figure 25: Test accuracies (y-axis) over timestep (x-axis) for FSOL-WRS with reservoir size $K = 64$ using exponential weighting on all 16 tested datasets. **Light grey lines:** test accuracies of the FSOL baseline methods at each timestep. **Solid black lines:** test accuracies of the "oracle" models, computed as the cumulative maximum of the FSOL baselines. **Blue:** corresponds to FSOL-WRS variants ensembled via simple averaging. **Red:** corresponds to FSOL-WRS variants ensembled via weighted averaging. **Dotted lines:** indicate whether voting-based zeroing was applied for additional sparsity.

43

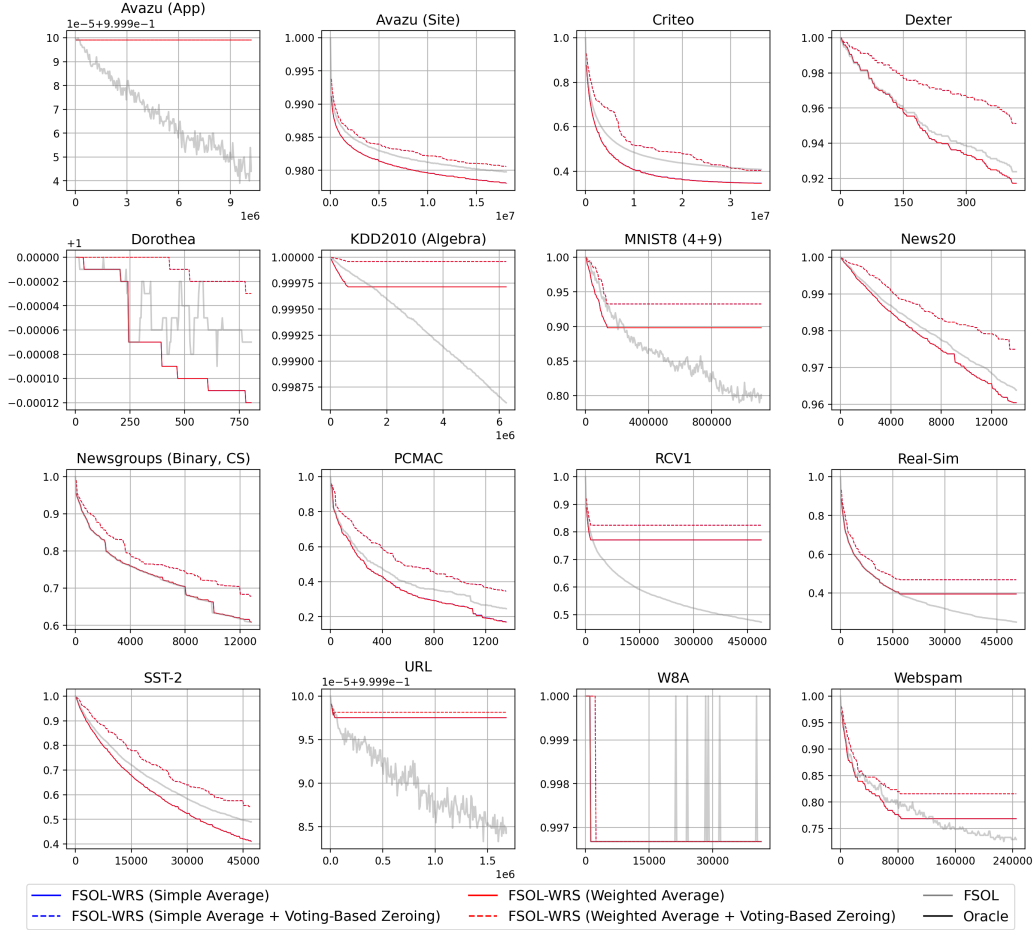



Figure 26: Sparsity (y-axis) over timestep (x-axis) for FSOL-WRS with reservoir size $K = 64$ using exponential weighting on all 16 tested datasets. **Light grey lines:** sparsities of the FSOL baseline methods at each timestep. **Blue:** corresponds to FSOL-WRS variants ensembled via simple averaging. **Red:** corresponds to FSOL-WRS variants ensembled via weighted averaging. **Dotted lines:** indicate whether voting-based zeroing was applied for additional sparsity.

44

G.2.2 PAC and PAC-WRS

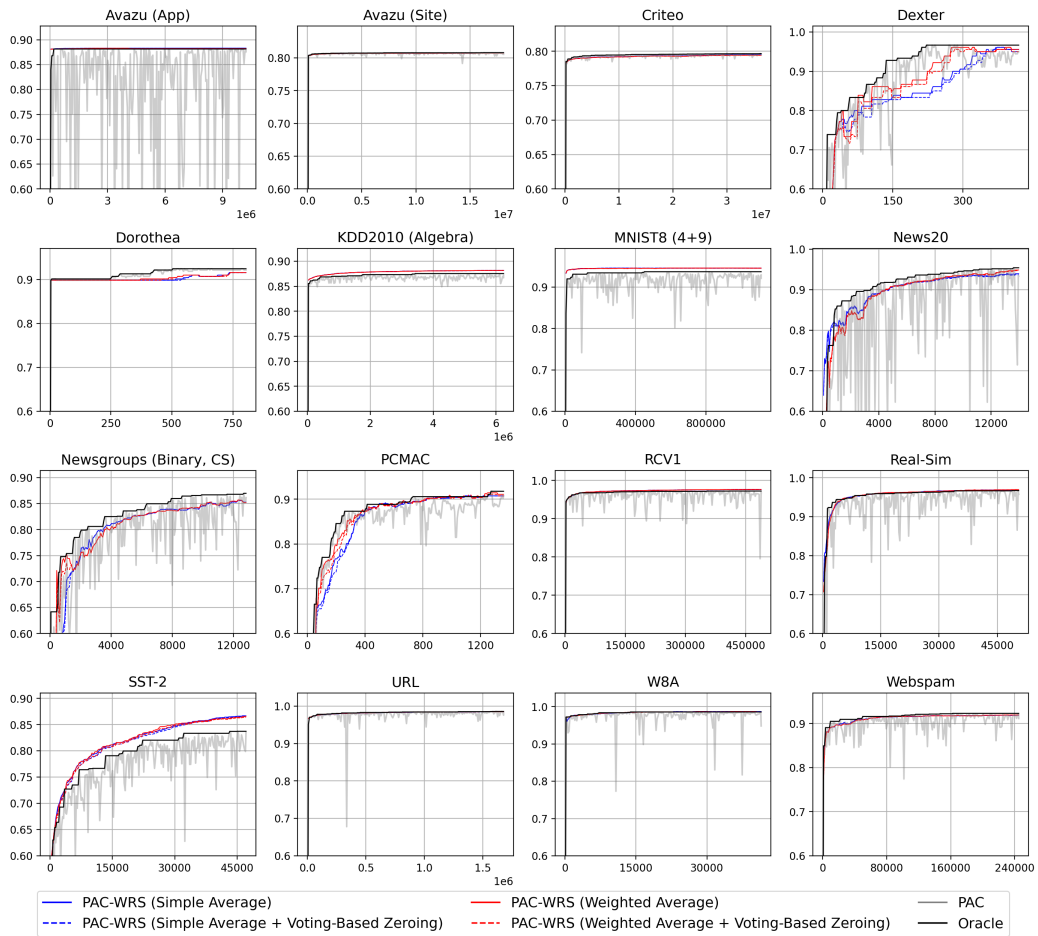



Figure 27: Test accuracies (y-axis) over timestep (x-axis) for PAC-WRS with reservoir size $K = 64$ using standard weighting on all 16 tested datasets. **Light grey lines:** test accuracies of the PAC baseline methods at each timestep. **Solid black lines:** test accuracies of the "oracle" models, computed as the cumulative maximum of the PAC baselines. **Blue:** corresponds to PAC-WRS variants ensembled via simple averaging. **Red:** corresponds to PAC-WRS variants ensembled via weighted averaging. **Dotted lines:** indicate whether voting-based zeroing was applied for additional sparsity.

45

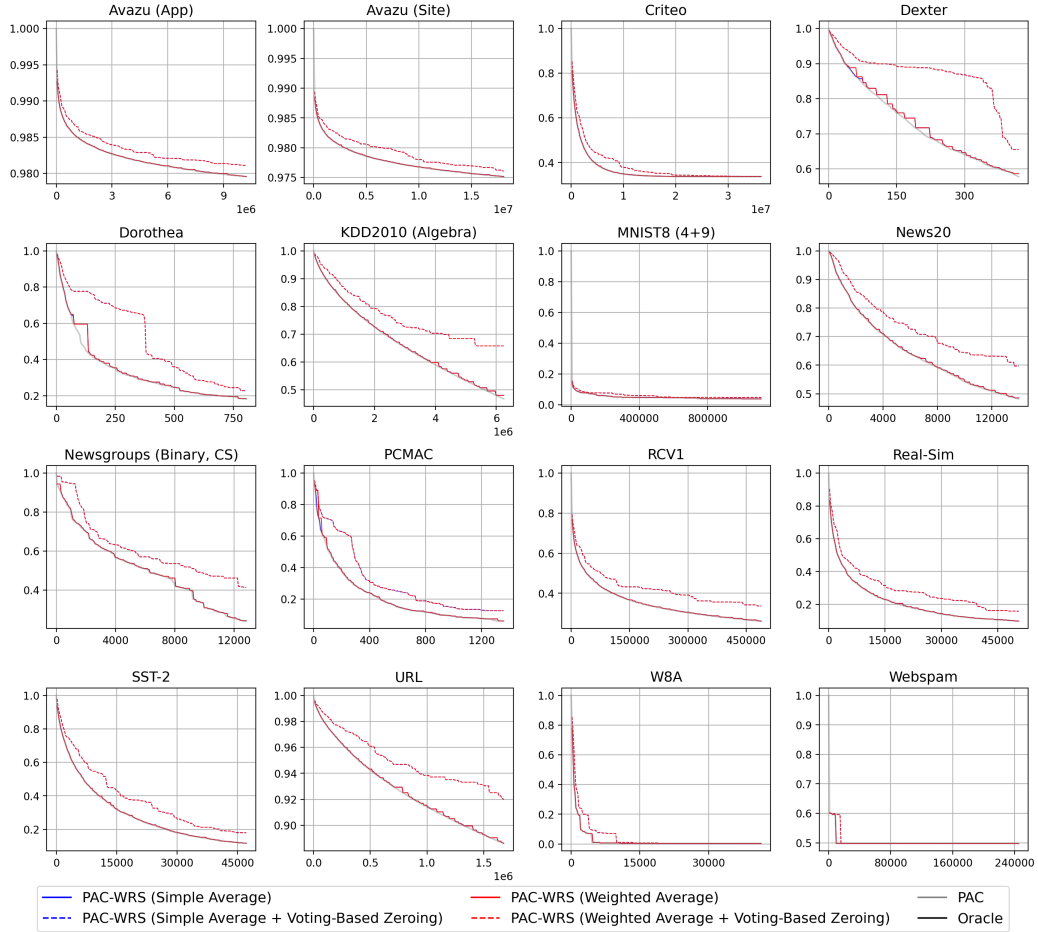



Figure 28: Sparsity (y-axis) over timestep (x-axis) for PAC-WRS with reservoir size $K = 64$ using standard weighting on all 16 tested datasets. **Light grey lines:** sparsities of the PAC baseline methods at each timestep. **Blue:** corresponds to PAC-WRS variants ensembled via simple averaging. **Red:** corresponds to PAC-WRS variants ensembled via weighted averaging. **Dotted lines:** indicate whether voting-based zeroing was applied for additional sparsity.

46

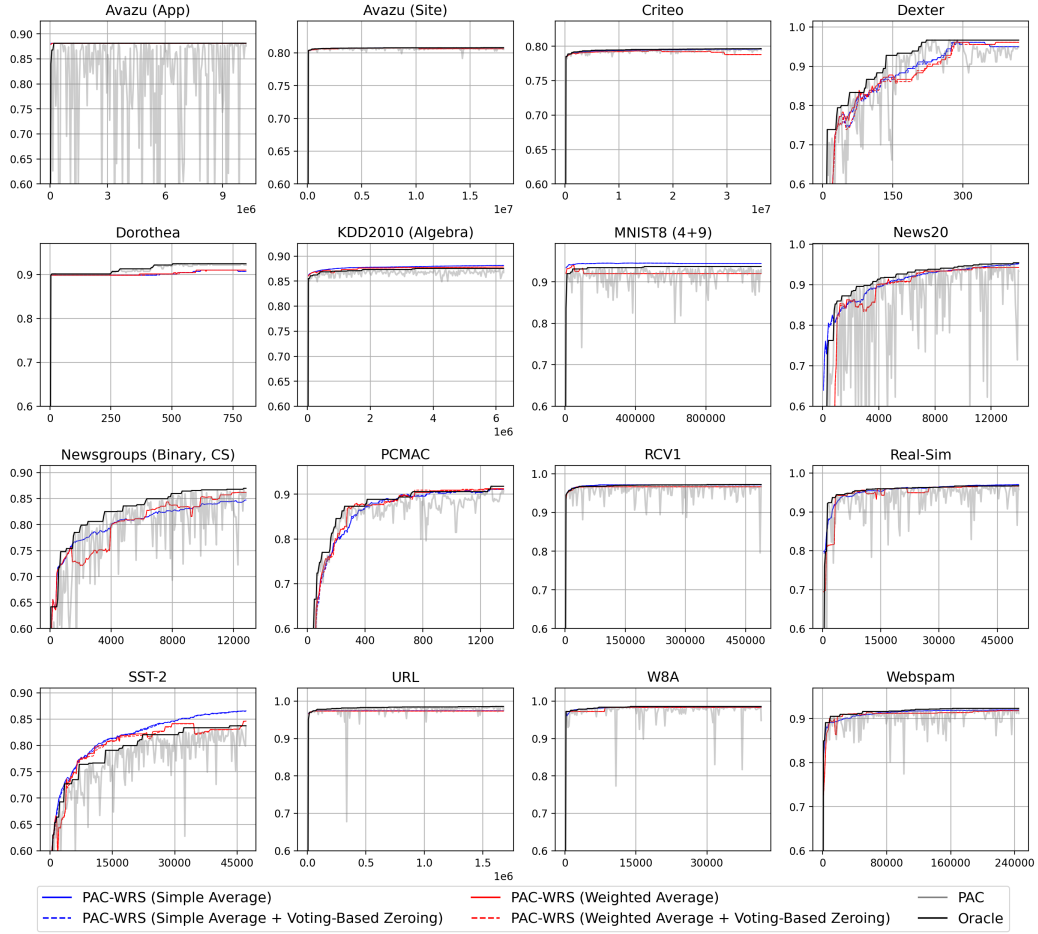



Figure 29: Test accuracies (y-axis) over timestep (x-axis) for PAC-WRS with reservoir size $K = 64$ using exponential weighting on all 16 tested datasets. **Light grey lines:** test accuracies of the PAC baseline methods at each timestep. **Solid black lines:** test accuracies of the "oracle" models, computed as the cumulative maximum of the PAC baselines. **Blue:** corresponds to PAC-WRS variants ensembled via simple averaging. **Red:** corresponds to PAC-WRS variants ensembled via weighted averaging. **Dotted lines:** indicate whether voting-based zeroing was applied for additional sparsity.

47

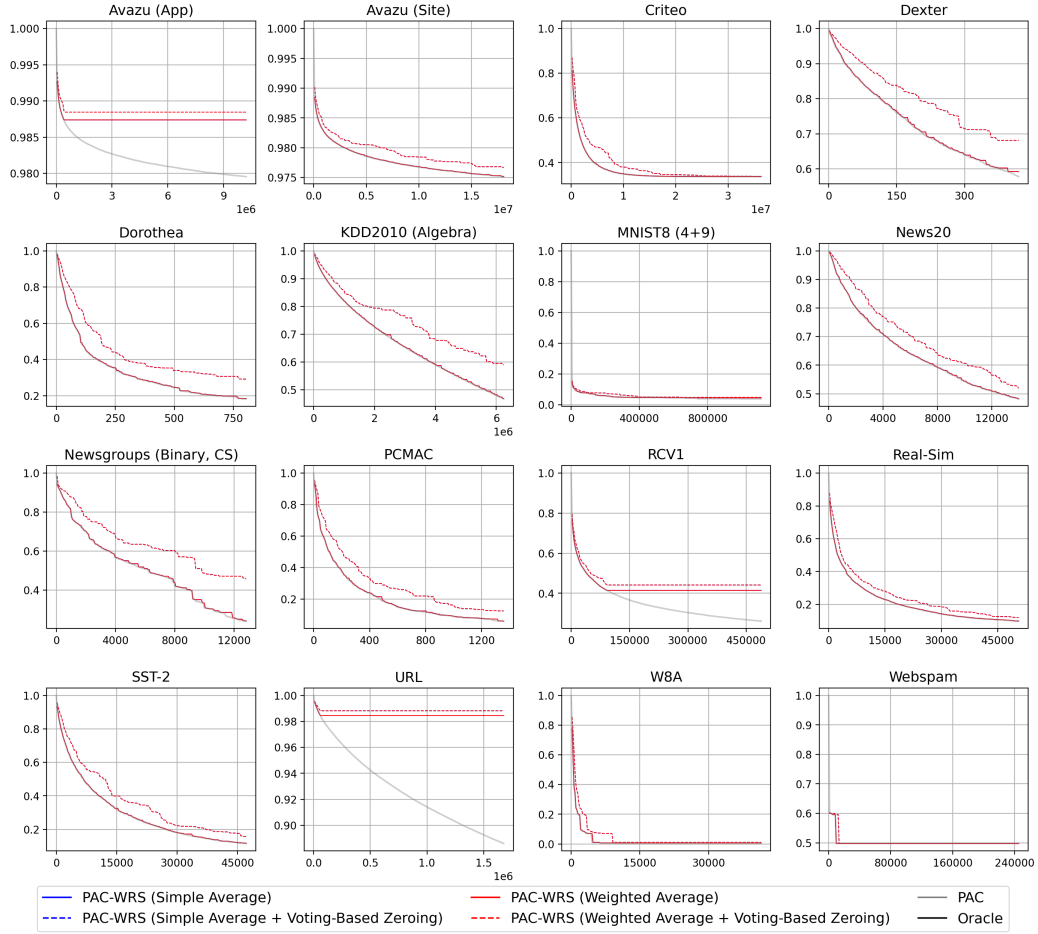



Figure 30: Sparsity (y-axis) over timestep (x-axis) for PAC-WRS with reservoir size $K = 64$ using exponential weighting on all 16 tested datasets. **Light grey lines:** sparsities of the PAC baseline methods at each timestep. **Blue:** corresponds to PAC-WRS variants ensembled via simple averaging. **Red:** corresponds to PAC-WRS variants ensembled via weighted averaging. **Dotted lines:** indicate whether voting-based zeroing was applied for additional sparsity.

48

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Please see our Abstract and Introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

49

Justification: Please see Section 3.2 and Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Please see Section 4 and Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

50

Answer: [Yes]

Justification: Please see our GitHub at https://github.com/FutureComputing4AI/Weighted-Reservoir-Sampling-Augmented-Training/tree/main, with a detailed `README.md` file.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please see Algorithm 1, Section 4, and Appendix B, as well as our GitHub at https://github.com/FutureComputing4AI/Weighted-Reservoir-Sampling-Augmented-Training/tree/main.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Please see error bars and significance tests in Section 4 and Appendix G.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

51

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Please see Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We have carefully read the NeurIPS Code of Ethics and have made sure that our research conforms to it.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This paper introduces a foundational research method that does not have any direct paths to negative societal impacts, aside from the general interactions of machine learning with society.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

52

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper does not involve any data or models that have a high risk for misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We enumerate and cite all datasets in Table 1. Aside from our own preprocessed versions of Newsgroups (Binary, CS) and SST-2, the 14 other datasets are all publicly available.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

53

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Please see the `README.md` file in our accompanying GitHub https://github.com/FutureComputing4AI/Weighted-Reservoir-Sampling-Augmented-Training/tree/main.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No crowdsourcing or work with human subjects was performed for this paper.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not have any study participants and/or crowdsourcing.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

54

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

55