Online Federated Learning via Non-Stationary Detection and Adaptation Amidst Concept Drift

Bhargav Ganguly, *Graduate Student Member, IEEE*, and Vaneet Aggarwal, *Senior Member, IEEE*

***Abstract*— Federated Learning (FL) is an emerging domain in the broader context of artificial intelligence research. Methodologies pertaining to FL assume distributed model training, consisting of a collection of clients and a server, with the main goal of achieving optimal global model with restrictions on data sharing due to privacy concerns. It is worth highlighting that the diverse existing literature in FL mostly assume stationary data generation processes; such an assumption is unrealistic in real-world conditions where concept drift occurs due to, for instance, seasonal or period observations, faults in sensor measurements. In this paper, we introduce a multiscale algorithmic framework which combines theoretical guarantees of *FedAvg* and *FedOMD* algorithms in near stationary settings with a non-stationary detection and adaptation technique to ameliorate FL generalization performance in the presence of concept drifts. We present a multi-scale algorithmic framework leading to $\tilde{\mathcal{O}}(\min\{\sqrt{LT}, \Delta^{(1/3)}T^{(2/3)}+\sqrt{T}\})$ *dynamic regret* for T rounds with an underlying general convex loss function, where L is the number of times non-stationary drifts occurred and Δ is the cumulative magnitude of drift experienced within T rounds.**

***Index Terms*— Federated learning (FL), non-stationary, dynamic regret.**

I. Introduction

ADVANCEMENTS in technology and science have led to an increase in raw data generation and processing power at smart devices, prompting the development of large-scale distributed machine learning architectures, such as Federated Learning (FL), which enables local training at end-user devices and periodic global model synchronization. However, most existing FL literature assumes time-invariant data generation processes [1]. In reality, data generation processes at end devices can be impacted by abrupt changes in the underlying environment (e.g., pandemic on flight booking data [2]). This paper models the non-stationary data generation environment and analyzes the *dynamic regret* of the proposed algorithms in this setup.

Manuscript received 18 November 2022; revised 6 May 2023; accepted 6 July 2023; approved by IEEE/ACM Transactions on Networking Editor G. Joshi. Date of publication 1 August 2023; date of current version 16 February 2024. This work was supported in part by Cisco Inc., Defense Advanced Research Projects Agency (DARPA), under Grant W911NF2020003 and in part by the National Science Foundation (NSF) under Grant CCF-2149588. *(Corresponding author: Vaneet Aggarwal.)*

Bhargav Ganguly is with the School of Industrial Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: bganguly@purdue.edu).

Vaneet Aggarwal is with the School of Industrial Engineering and the School of ECE, Purdue University, West Lafayette, IN 47907 USA, and also with the CS Program, KAUST, Thuwal 23955, Saudi Arabia (e-mail: vaneet@purdue.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2023.3294366, provided by the authors.

Digital Object Identifier 10.1109/TNET.2023.3294366

In FL literature, such non-stationary behavior observed in the data generation model over time is commonly called *concept drift* [3], [4]. We note that such a phenomenon is also called *covariate shift* in the wider context of general machine learning [5]. Conventional FL methodologies such as the *FedAvg* [6] being agnostic to such time varying data shifts end up producing worse generalization results especially on ML classification/regression tasks. Hence, it is critical to augment such learning frameworks with non-stationarity detection and adaptation procedures to mitigate staleness/poor generalization ability of obtained ML models. Most of the existing work in non-stationary FL literature leverage heuristic techniques to ensure model robustness in the midst of drifts, including sliding window based adaptive learning [7], ensemble learning [8], and regularization mechanisms [9]. Although the problem has been widely studied for algorithms, this paper provides the first results on *dynamic regret* for online convex optimization for general convex functions in the FL setup. We note that online convex optimization has been studied in dynamic environments for both centralized and distributed settings (See Sec. II for detailed comparison). We note that the methodologies proposed for centralized learning in non-stationary environments leverage the convenience of having the exact knowledge of both newly collected datasets and models at every learning round which is not available in federated learning. Furthermore, raw data offloading by the clients is restricted in FL due to privacy concerns.

We highlight that the key novelty of this work is an efficient *drift detection and adaptation* method that is suitably augmented with a randomized baseline FL algorithm scheduling procedure and facilitates *training at multiple scales*. To demonstrate what our framework accomplishes at a high level of abstraction, let us consider a scenario where FL training is desired to be conducted over $T = 7$ with *FedAvg*. Our proposed framework will equip clients to train over shorter allowable time horizons in powers of 2, i.e., $\{1, 2, 4\}$ for the current example, with suitable learning rates for each such time chunks, i.e., *FedAvg* $\{\frac{1}{\sqrt{1}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{4}}\}$ for horizons of lengths 1,2,4, respectively. We highlight that our framework carefully randomizes how such shorter training chunks are scheduled and it is not necessary that every allowable chunk is included for training. Furthermore, for the current example, during each round $t \in \{1, 2, \cdots, 7\}$, additional non-stationary tests that involve FL training loss collected by the clients are introduced in our method to identify and adaptively train for *concept drift*.

In the context of ensuing discussion on our proposed methodology, the construction of the aforementioned pieces: *drift detection and adaptation*, and *multiscale learning* are in fact closely tied to the *near-stationarity* properties of the baseline FL algorithms. More specifically, we demonstrate mathematically why baseline FL algorithms may not directly ensure optimal learning in high degrees of drifts. Along these lines, we further show how their favorable behavior in *near-stationary* regime may be leveraged to derive a supplementary quantity that can support drift aware learning in a multiscale fashion. In the later sections, we explicitly delineate how the key components of our algorithmic framework are concretely established based on the aforementioned mathematical ideas and bring out the connections more explicitly in the theoretical analysis of *dynamic regret* incurred by our algorithm. In summary, the major contributions of our work are as follows:

1) We propose a multi-scale algorithmic framework which can equip any existing baseline FL methodologies that work well in *near-stationary* environments with a suitably designed change detection and adaption technique to mitigate model staleness in high drifting learning environments. More specifically, our current methodology is a general unified framework which is shown to be directly compatible with two widely deployed baseline FL algorithms: *FedAvg* [6] and *FedOMD* [15].
2) We conduct comprehensive theoretical analysis of the proposed framework and characterize the performance by *dynamic regret* over all the FL rounds in terms of both the accumulated magnitude of such drifts (denoted by Δ, see Definition 1), and the number of times data drifts have occurred (denoted by L, see Definition 2). Mathematically, we show that adopting our approach leads to bounding the *dynamic regret* over T FL rounds with a general convex underlying loss measure as $\tilde{\mathcal{O}}(\min\{\sqrt{LT}, \Delta^{\frac{1}{3}}T^{\frac{2}{3}} + \sqrt{T}\})$.
3) We provide proof-of-concept experimental evaluations on ML image classification datasets bolstering the efficacy of the proposed framework, wherein we compare the performance with two other widely-used competing FL algorithms.

We organize rest of the paper as summarized next. In Section II, we present a comparison of our results with prior works in centralized and distributed non-stationary optimization. In Section III, we formally describe the problem of *dynamic regret* minimization for Federated Learning in drifting environments, whereby we also introduce key assumptions and definition necessary for our algorithm development. In Section IV, we explain the various components and sub-routines associated with our multi-scale Federated Learning methodology. Furthermore, we present rigorous analysis as well as elucidation of the obtained mathematical results for the proposed algorithm leading to the aforementioned *dynamic regret* bound in Section V. In Section VI, we report our simulation results investigating how our framework behaves in practice when deployed on real-world datasets under various drift scenarios.

II. Related Work on Online Convex Optimization in Dynamic Environments

For centralized convex optimization in dymanic environments, several existing works propose online algorithms with *dynamic regret* bounds in terms of problem-specific quantities. In this regard, three very commonly used metrics are: *comparator regularity* accumulating the changes in the minimizers over T horizon, i.e., C_T; *temporal variability* capturing the differences in the loss function values, i.e., Δ; and *gradient variability* which tracks loss in gradients instead of actual function lossses, i.e., D_T (see definition of C_T, D_T in Appendix A of the Supplementary Material). In our problem setting, the usage of Δ (see Definition 1) is inline with mathematical formulations of drift in recent FL literature [3], [16], and is direct adoption of *temporal variability* to a distributed learning setting. Furthermore, we replace C_T with a more intuitive drift measure L, i.e., the exact number of times drift occurred over horizon T. Previous works based on adaptive online gradient descent [10], online mirror descent [11] are centralized counterparts to our setting; and produce a *dynamic regret* of $\tilde{\mathcal{O}}(\sqrt{T}(C_T+1))$ and $\tilde{\mathcal{O}}(\min\{\sqrt{D_T+1} + \sqrt{(D_T+1)C_T}, (D_T+1)^{1/3}T^{1/3}\Delta^{1/3}\})$, respectively. On the other hand, we provide a general *dynamic regret* bound of $\tilde{\mathcal{O}}(\min\{\sqrt{LT}, \Delta^{\frac{1}{3}}T^{\frac{2}{3}} + \sqrt{T}\})$ with the flexibility of choosing any baseline FL optimizer which can guarantee worst case $\tilde{\mathcal{O}}(\sqrt{T})$ regret in any *near-stationary* environment. Furthermore, we note that the strategies for centralized setup do not directly provide the *dynamic regret* guarantees in the distributed/federated scenarios since the distributed/federated setup is more challenging as the data collection and learning process is spread across a network of client devices.

In the distributed setting, an online mirror descent algorithm was proposed in [12] achieving a $\tilde{\mathcal{O}}(\sqrt{T(C_T+1)})$ assuming prior knowledge of C_T and T. The authors of [13] present an online gradient tracking methodology; thereby obtaining a $\tilde{\mathcal{O}}(\sqrt{(C_T+1)}T^{3/4})$ with no prior knowledge of the environment dynamics. Furthermore, the aforementioned bound was improved by introducing *gradient variability* D_T in conjunction with C_T to $\tilde{\mathcal{O}}(\sqrt{TC_T} + \sqrt{T} + D_T + \sqrt{C_T D_T})$ in [14]. We note that when $C_T = O(T^{1-\epsilon})$ for small ϵ, the bound in [13] is larger than $\tilde{\mathcal{O}}(T)$, making this bounds not interesting in large drift scenarios, since in our case when $L = O(T^{1-\epsilon})$, the bound is $\tilde{\mathcal{O}}(T^{1-\epsilon/2})$ which is sub-linear. Further, the bound in [14] is in terms of gradient changes, while our bound is independent of gradient changes. More specifically, we highlight that the bounds revealed in terms of *gradient variability* measure D_T in [11] and [14] are direct consequences of additional Lipschitz smoothness assumption on loss gradients, whereas we keep our analysis restricted to general bounded, Lipschitz smooth convex loss measures.

We note that *dynamic regret* bounds could be improved to $\tilde{\mathcal{O}}(1+C_T)$ when loss functions are strongly convex as reported in recent studies pertaining to distributed ML [17], [18]. However, our proposed methodology relies only on the underlying loss measure being general convex, and therefore is not directly comparable to the aforementioned studies requiring strong convexity assumptions.

To the best of our knowledge, our algorithm design and consequent theoretical findings in FL setting extends the aforementioned prior bounds in the distributed paradigm where the underlying loss function is general convex, while it introduces tighter bounding terms using the metric L representing the

TABLE I

Dynamic Regret BOUNDS FOR CENTRALIZED, DISTRIBUTED, AND FEDERATED ONLINE CONVEX OPTIMIZATION METHODOLOGIES. FOR THE KEY ASSUMPTIONS, WE USE THE FOLLOWING SHORTENED DESCRIPTIONS : BLCL → BOUNDED, LIPSCHITZ, CONVEX LOSS FUNCTION; LG → LIPSCHITZ GRADIENT

References	Problem setting	Non-Stationary Measures	Prior Knowledge	Key Assumptions	*Dynamic Regret* Bound
[10]	Centralized	C_T	No	BLCL	$\tilde{\mathcal{O}}(\sqrt{T}(C_T+1))$
[11]	Centralized	C_T, D_T, Δ	No	BLCL, LG	$\tilde{\mathcal{O}}(\min\{\sqrt{D_T+1}+\sqrt{(D_T+1)C_T}, (D_T+1)^{1/3}T^{1/3}\Delta^{1/3}\})$
[12]	Distributed	C_T	Yes	BLCL	$\tilde{\mathcal{O}}(\sqrt{T(C_T+1)})$
[13]	Distributed	C_T	No	BLCL	$\tilde{\mathcal{O}}(\sqrt{(C_T+1)}T^{3/4})$
[14]	Distributed	C_T, D_T	No	BLCL, LG	$\tilde{\mathcal{O}}(\sqrt{TC_T}+\sqrt{T}+D_T+\sqrt{C_TD_T})$
This work	Federated	L, Δ	No	BLCL	$\tilde{\mathcal{O}}(\min\{\sqrt{LT}, \Delta^{\frac{1}{3}}T^{\frac{2}{3}}+\sqrt{T}\})$

number of changes instead of C_T. Furthermore, we note that although our algorithmic framework is designed for federated optimization, it can be straightforwardly extended to a distributed learning setup with consensus-based model averaging. More specifically, our results are indifferent to how model averaging is conducted across the clients, therefore, will hold in general distributed convex optimization in dynamic environments. The comparison of *dynamic regret* results discussed both in the centralized/distributed regime discussed thus far are summarized in Table I.

To emphasize, our work is well-aligned with dynamic federated optimization problem setting with the following high-level attributes: (i) it can combine any conventional baseline FL optimizer that works well in a *near stationary* setting with a multi-scale procedure that can detect and adapt in highly dynamic environments, and (ii) to the best of our knowledge, our framework is guaranteed to produce tighter sublinear *dynamic regret* bounds for a more generic set of non-stationarity measures L, Δ and runtime horizon T. Prior knowledge of L, Δ, and T are not required.

III. Problem Formulation

In this section, we formally put forth the mathematical problem aiming towards *dynamic regret* minimization for FL over client devices with time-varying data drifts. In the following discussion, we also denote the client devices as Data Processing Units (DPUs) since they conduct local ML training, as well as to clearly differentiate from the central-coordinator server which only performs model syncronization and training orchestration tasks. We collectively denote the set of DPUs as $\mathcal{N}$, and the central server as $\mathcal{S}$. Furthermore, we assume that FL training and model syncronization is performed over $t = 1, 2, \cdots, T$ rounds. Further, we denote the dataset generated at each DPU $n \in \mathcal{N}$ during rounds $t = 1, 2, \cdots, T$ as $\mathcal{D}_n^{(t)}$. For the ease of our analysis and presentation of results, we define the following quantities:

$$D_n^{(t)} = |\mathcal{D}_n^{(t)}|, \quad D^{(t)} = \sum_{n\in\mathcal{N}} D_n^{(t)}, \quad p_n^{(t)} = \frac{D_n^{(t)}}{D^{(t)}}. \tag{1}$$

At rounds $t = 1, \cdots, T$, each DPU $n \in \mathcal{N}$ is associated with a *local loss* function $F_n^{(t)}(\mathbf{x})$. It locally computes the loss gradient at the current global parameter vector $\mathbf{x}^{(t-1)}$, i.e., $\nabla F_n^{(t)}(\mathbf{x}^{(t-1)})$. For the ML model vector, we assume that $\mathbf{x} \in \mathbb{R}^d$. Formally,

$$F_n^{(t)}(\mathbf{x}) = \frac{1}{D_n^{(t)}} \sum_{\xi\in\mathcal{D}_n^{(t)}} f(\mathbf{x};\xi), \tag{2}$$

$$\nabla F_n^{(t)}(\mathbf{x}^{(t-1)}) = \frac{1}{D_n^{(t)}} \sum_{\xi\in\mathcal{D}_n^{(t)}} \nabla f(\mathbf{x};\xi)\big|_{\mathbf{x}=\mathbf{x}^{(t-1)}}, \tag{3}$$

where $f(\cdot\ ;\ \cdot)$ is the underlying ML loss function. Subsequently, all DPUs $n \in \mathcal{N}$ locally update their local ML model, i.e., $\mathbf{x}_n^{(t)}$. We note that this update procedure is the key difference across different conventional FL algorithms. We call this update procedure as `FL-UPDATE(·)`. Hence, we have:

$$\mathbf{x}_n^{(t)} = \texttt{FL-UPDATE}\big(\mathbf{x}^{(t-1)}, \nabla F_n^{(t)}(\mathbf{x}^{(t-1)})\big). \tag{4}$$

In this regard, we highlight that the explicit formulation of `FL-UPDATE(·)` for *FedAvg, FedOMD* has been provided in Section V-A.

Subsequently, each DPU $n \in \mathcal{N}$ communicates its locally updated model, i.e., $\mathbf{x}_n^{(t)}$ to the central aggregation server which we denote by $\mathcal{S}$. $\mathcal{S}$ then performs the aggregation step to obtain the global ML model, i.e., $\mathbf{x}^{(t)}$ for each t, according to the following:

$$\mathbf{x}^{(t)} = \sum_{n\in\mathcal{N}} p_n^{(t)} \mathbf{x}_n^{(t)}. \tag{5}$$

After the aggregation step, this $\mathbf{x}^{(t)}$ is sent back by $\mathcal{S}$ to DPUs in $\mathcal{N}$ for conducting next round of local ML training. This process of local training and periodic aggregation is repeated across all the federated learning rounds $t \in [1, T]$.

Remark 1: Often a small number of model parameters are updated locally for any client DPU in a FL learning setting, and usually the learnt ML models (for instance DNN-based image classifiers) are large-scale training architectures which potentially induce communication bottlenecks in the network. In such cases, exchanging model differentials instead of the actual model itself ameliorates communication overhead significantly. Our framework can directly accommodate such communication efficient message passing setups that involve model differentials (as detailed in Appendix B, see the Supplementary Material).

Next, we elaborate on the essential quantities that are heavily utilized throughout our theoretical analysis pertaining to online *dynamic regret*. In this context, we first highlight that we are concerned with the ML losses incurred by the DPUs at the end of each round of FL training and global aggregation. More precisely, the ML model $\mathbf{x}^{(t)}$ updated during rounds $t = 1, 2, \cdots, T$ is applied on the observed datasets $\mathcal{D}_n^{(t)}$ to collect local loss $F_n^{(t)}(\mathbf{x}^{(t)})$ as defined in Eq. (2). The local ML losses are combined according to the proportion of dataset sizes thereby obtaining the global losses as follows:

$$F^{(t)}(\mathbf{x}^{(t)}) = \sum_{n\in\mathcal{N}} p_n^{(t)} F_n^{(t)}(\mathbf{x}^{(t)}). \tag{6}$$

Finally, in our system model, the objective of the network comprised by DPUs in $\mathcal{N}$ and server $\mathcal{S}$ is to minimize the global *dynamic regret* $R_{[1,T]}$ given by:

$$R_{[1,T]} = \sum_{t=1}^{T} F^{(t)}(\mathbf{x}^{(t)}) - \sum_{t=1}^{T} F^{(t)}(\mathbf{x}^{(t),*}), \tag{7}$$

where $\mathbf{x}^{(t),*} = \arg\min_{\mathbf{x}\in\mathbb{R}^d} F^{(t)}(\mathbf{x})$. We now introduce the key assumptions and definitions pertaining to loss functions and baseline FL methods that will be used throughout the course of our analysis.

Assumption 1 (Convexity, Smoothness and Boundedness of ML Loss Function): We assume that the underlying ML loss function, i.e., $f(\cdot;\xi)$ is convex, bounded in $[0,1]$ and μ-Lipschitz w.r.t. $\|\cdot\|$, which implies the following $\forall \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^d$:

$$f(\mathbf{x};\xi) - f(\tilde{\mathbf{x}};\xi) \le \mu\|\mathbf{x} - \tilde{\mathbf{x}}\|. \tag{8}$$

We highlight that Assumption 1 is general enough and captures a wide range of problems pertaining to Machine Learning and Online Optimization tasks.

Definition 1 (Concept Drift): The online Concept Drift between two consecutive rounds of global aggregation $t-1$ and t is measured by $\Delta_t \in \mathbb{R}^+$, which captures the maximum variation of the global loss function for any arbitrary ML model $\mathbf{x} \in \mathbb{R}^d$ according to

$$|F^{(t)}(\mathbf{x}) - F^{(t-1)}(\mathbf{x})| \le \Delta_t. \tag{9}$$

Further, the cumulative concept drift incurred across rounds $t_1, t_1+1, \cdots, t_2$ is denoted by:

$$\Delta_{[t_1,t_2]} \triangleq \sum_{\tau=t_1}^{t_2} \Delta_\tau. \tag{10}$$

Definition 2 (Number of Non-Zero Distribution Shift Events): Using the notion of online concept drift Δ in Definition 1, we quantify the number of FL rounds with non-zero drift across $t = 1, 2, \cdots, T$, and denote it by L. Mathematically,

$$L = \sum_{t=1}^{T} \mathbb{1}[\Delta_t \ne 0]. \tag{11}$$

The non-stationary measures L, Δ quantify the degree of time-varying nature of the datasets generated at the DPUs. Hence, larger drifts should directly imply more frequent calibration (i.e., change detection and adaptation) to achieve staleness resiliency for learnt ML models. We note that the existing methodologies for drift management include both local drift, where the statistical properties of the data distribution that individual clients experience over time [19], and global drift, where the statistical properties of the data distribution across all the clients over time [4], [20]. In this paper, we consider global drift, while the results would be applicable even for the local drift. Next, we define the functions $\rho(\cdot), C(\cdot)$ which we later use to mathematically characterize the notion of *near-stationarity*.

Definition 3: We define a non-increasing function $\rho : [t] \rightarrow \mathbb{R}$ such that $\rho(t) \ge \frac{1}{\sqrt{t}}$, and $C(t) = t\rho(t)$ is an increasing function of t. Furthermore, for a given T, we define $\hat{\rho}(t) = 6(\log_2 T + 1)\log(T/\delta)\rho(t)$.

In our framework, we require the underlying baseline FL algorithms to have certain guarantees in environments where the cumulative drift is small. We denote such environments as *near-stationary*. This requirement from baseline algorithms in *near-stationary* settings is expressed via Assumption 1 which is presented next.

Requirement 1 (Base Algorithm Performance Guarantee in a Near-Stationary Environment): We assume that the base algorithm produces an auxiliary quantity $\tilde{F}^{(t)}$ at the end of each global round of aggregation $t \in \{1, 2, \cdots, T\}$ satisfying the following:

$$\tilde{F}^{(t)} \le \max_{\tau\in[1,t]} F^{(\tau)}(\mathbf{x}^{(\tau),*}) + \Delta_{[1,t]}, \tag{12}$$

$$\frac{1}{t}\sum_{t=1}^{t}[F^{(t)}(\mathbf{x}^{(t)}) - \tilde{F}^{(t)}] \le \rho(t) + \Delta_{[1,t]}. \tag{13}$$

where $\rho(.)$ is described in Definition 3. Further, $\Delta_{[1,t]}$ represents the cumulative concept drift experienced with $\Delta_{[1,t]} \le \rho(t)$, i.e., near-stationary environment.

We emphasize that this supplementary estimator $\tilde{F}^{(t)}$ is not a direct output of most conventional FL algorithms. However, the construction of this quantity is possible from the models learnt over time. Specifically, we outline the exact form of this quantity and verify the correctness of Assumption 1 for conventional baseline FL algorithms: *FedAvg*, *FedOMD* in Appendix E, see the Supplementary Material.

Furthermore, a key intuition behind Requirement 1 is the idea a single instance of a vanilla baseline FL algorithm should be enough when environment is *near-stationary*, i.e., $\Delta_{[1,t]} \le \rho(t)$. However, with increased degree of non-stationarity i.e., $\Delta_{[1,t]} > \rho(t)$, the conditions stated in Requirement 1 should be somehow proxied to detect drifts and restart learning. We demonstrate this key intuition with mathematical justifications in Appendix F, see the Supplementary Material.

In our subsequent discussion delineating the algorithmic framework for non-stationary FL, we explicitly demonstrate how this auxiliary quantity is useful to design suitable drift detection tests (**Test 1** and **Test 2** in Algorithm 3, Section IV) that attempt to proxy the conditions of Requirement 1, thereby promoting multi-scale learning in dynamic environments.

IV. Algorithmic Framework

In this section, we provide an elaborate description of our multi-scale algorithmic framework to perform FL in dynamic environments. In this regard, we note that we adopt the notions of multi-scale base algorithm initializations and drift detection mechanism for FL from [21], which was originally proposed for non-stationary reinforcement learning training. Roughly speaking, in our framework, multiple base FL instances are scheduled, and in turn, are equipped with a carefully engineered change detection mechanism to mitigate model staleness. These base FL instances are conventional FL algorithms that satisfy Assumption 1, we specifically show in Appendix E, see the Supplementary Material, that this Assumption holds for *FedAvg* and *FedOMD*.

Algorithm 1 Randomized Scheduling Procedure

```
1:  Input: $m$, $\rho(\cdot)$, First round $t$.
2:  Output: A collection of base FL instances $\boldsymbol{A}$
3:  Initialize: $\boldsymbol{A} = \emptyset$.
4:  for $\tau = t, t+1, \cdots, t+2^m-1$ do
5:    for $k = m, m-1, \cdots, 0$ do
6:      if $\tau \bmod 2^k = 0$ then
7:        Schedule $\mathcal{A} := (\mathcal{A}.s, \mathcal{A}.e, \eta^A, \mathbf{x}^{\mathcal{A}})$ w.p. $\frac{\rho(2^m)}{\rho(2^k)}$.
8:        Update Instance set : $\boldsymbol{A} = \boldsymbol{A} \cup \mathcal{A}$.
9:      end if
10:   end for
11: end for
```

Algorithm 2 Multi-Scale FL Runner (MSFR)

```
1: Input: $m$, $\rho(\cdot)$, FL round $t$, RunScheduler
2: Output: Current Loss $F^{(t)}$, Optimistic FL Loss estimate $\tilde{F}^{(t)}$.
3: if RunScheduler = True then
4:   $\boldsymbol{A} \leftarrow$ Randomized Scheduling Procedure$(m, \rho(\cdot), t)$.
5:   RunScheduler $\leftarrow$ False.
6: end if
7: At round $t$, $\mathcal{A}_t = \arg\min_{\mathcal{A} \in \boldsymbol{A}} \mathcal{A}.e - t$ to be run at all DPUs.
8: Receive $\{F^{(t)}, \tilde{F}^{(t)}\}$ from $\mathcal{A}_t$; $F^{(t)}$ is on model $\mathbf{x}^{(t),\mathcal{A}_t}$.
```

Next, we present Randomized Scheduling Procedure in Algorithm 1. An instance $\mathcal{A} = (\mathcal{A}.s, \mathcal{A}.e, \eta^A, \mathbf{x}^{\mathcal{A}})$ is desired to be scheduled over the horizon $[\mathcal{A}.s, \mathcal{A}.e]$. $\eta^A = \frac{1}{\sqrt{\mathcal{A}.e - \mathcal{A}.s + 1}}$ is the learning rate for $\mathcal{A}$ over the scheduled horizon $[\mathcal{A}.s, \mathcal{A}.e]$ and $\mathbf{x}^{\mathcal{A}}$ is the corresponding model learnt over the aforementioned horizon for instance $\mathcal{A}$. We also provide an elaborate mathematical reasoning for the choices of learning rates for *FedAvg*, *FedOMD* in Section V. Furthermore, we highlight that all the scheduling orchestrations and change detection procedure are conducted via server $\mathcal{S}$, whereas model training is still locally conducted by the client DPUs, i.e., $\mathcal{N}$.

Algorithm 1 acts as a subroutine that can schedule multiple instances of base algorithm at different orders of scale in a carefully constructed randomized fashion. It populates a set of base FL instances, i.e., $\boldsymbol{A}$ with run lengths upto input order m and scheduled to start at different time periods for a particular block of rounds, i.e., $[t, t+2^m-1]$. Next, we present a more formal definition of *maximum scheduling order* input m for Algorithm 1.

Definition 4 (Maximum Scheduling Order m): The maximum order for which Randomized Scheduling Procedure (Algorithm 1) can be executed over an arbitrary interval $\mathcal{I} = [t_1, t_2]$ is given by $m_{\mathcal{I}} = \lceil \log_2(t_2 - t_1 + 1) \rceil$.

Next, in Algorithm 1, scheduling of base FL instances only happens with certain probability, i.e., $\frac{\rho(2^m)}{\rho(2^k)}$ for a certain order-k time block. The scheduling procedure is executed at the FL central server S, and subsequently communicated to all the DPUs in $\mathcal{N}$. It is necessary to emphasize that each DPU $n \in \mathcal{N}$ follows the same base algorithm schedule, however it conducts training only on its own local dataset $\mathcal{D}_n^{(t)}$ and updates local model to $\mathbf{x}_n^{(t)}$.

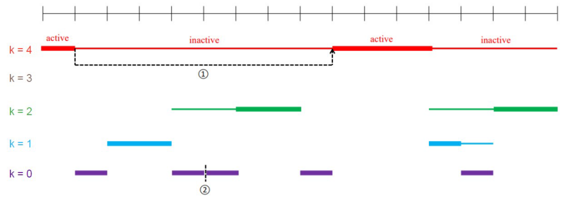


Fig. 1. Master-FL execution example with $m = 4$.

The aforementioned Randomized Scheduling Procedure (Algorithm 1) is only triggered via Multi-Scale FL Runner (Algorithm 2). Multi-Scale FL Runner (Algorithm 2) is dedicated to perform two tasks: scheduling base algorithms at the client DPUs requested via `RunScheduler` input flag, and decide the correct base instance run at the DPUs. Clearly, such an orchestration mechanism requires intervention of the server node $\mathcal{S}$. From $\mathcal{S}$, all the DPUs in $\mathcal{N}$ are directed to run the base instance with the *shortest remaining run length* whose schedule overlap with current FL round t in step 7 of Algorithm 2. We denote this instance as $\mathcal{A}_t$. We illustrate this greedy selection procedure for the baseline algorithm instances via an example presented next.

In Figure 1, we demonstrate the execution of different scheduled instances based on the shortest remaining length greedy rule in step 7 of Algorithm 2. We provide a toy example with $m = 4$ to illustrate how this rule works, as shown in [21]. As depicted in Figure 1, Algorithm 1 schedules one instance of length 2^4 (in red), two instances of length 2^2 (in green), two instances of length 2^1 (in blue), and five instances of length 1 (in purple). The bold sections of the lines represent the active periods of each instance. Furthermore, consider that the base instances are *FedAvg*. This implies the learning rates are $\frac{1}{\sqrt{2^0}}, \frac{1}{\sqrt{2^1}}, \frac{1}{\sqrt{2^2}}, \frac{1}{\sqrt{2^4}}$ for the various order $k = 0, 1, 2, 4$ instances respectively. Clearly, the order-4 instance ran for the first round, and then was paused for the next 8 rounds. Subsequently, it was executed for 3 more rounds before it was paused. The dashed line marked as ① means that learning restarts at round 9 for the order-4 instance with the learnt model at the end of round 1. While, the dashed line marked as ② represents that 2 order-0 instances were scheduled consecutive to each other. This means that although the two instances are successive, but learning happens from scratch in both the instances.

Consequently in step 8 of Algorithm 2, each DPU n, picks its model $\mathbf{x}_n^{(t-1),\mathcal{A}_t}$, performs model update according to Eq. (4), and shares $\{\mathbf{x}_n^{(t),\mathcal{A}_t}, F_n^{(t)}(\mathbf{x}_n^{(t),\mathcal{A}_t})\}$ with the central server $\mathcal{S}$. Finally, $\mathcal{S}$ computes and stores the instantaneous $\{F^{(t)}, \tilde{F}^{(t)}\}$.

Remark 2 (Computation of $F^{(t)}$): To enforce protection of sensitive data while DPU's share their local losses $\{F_n^{(t)}\}_{n \in \mathcal{N}}$ to server $\mathcal{S}$ for computation of global loss measure $F^{(t)}$, privacy-preserving techniques, such as differential privacy, homomorphic encryption or secure multi-party computation, can be deployed for loss function computations in a privacy-preserving manner [22], while a detailed investigation is left as future work.

Algorithm 3 Master-FL

```
1:  Input: $\hat{\rho}(\cdot)$ (see Definition 3).
2:  Initialize: $t \leftarrow 1$.
3:  for $m = 0, 1, \ldots$ do
4:      Set $t_{new} \leftarrow t$, RunScheduler ← True.
5:      Test-1-Flag ← 1, Test-2-Flag ← 1.
6:      while $t < t_{new} + 2^m$ do
7:          $\{F^{(t)}, \tilde{F}^{(t)}\} \leftarrow$ MSFR($m, \rho(\cdot), t$, RunScheduler).
            // Local training at DPUs $\mathcal{N}$
8:          Set $U_t = \max_{\tau \in [t_{new}, t]} \tilde{F}^{(t)}$
9:          if Test-1-Flag = 0 or Test-2-Flag = 0 then
10:             RunScheduler ← True.
11:             Break.
12:         end if
13:     end while
14: end for
15: Test 1: Current $\mathcal{A}_t$ is some order $k$ base instance.
16: if $t = \mathcal{A}_t.e$ then
17:     if $U_t \geq \sum_{\tau=\mathcal{A}.s}^{\mathcal{A}.e} F^{(t)} + 9\hat{\rho}(2^k)$ then
18:         Test-1-Flag ← 0.
19:     end if
20: end if

21: Test 2:
22: if $\frac{1}{t - t_{new} + 1} \sum_{\tau=t_{new}}^{t} [F^{(t)} - \tilde{F}^{(t)}] \geq 3\hat{\rho}(t - t_{new} + 1)$
    then
23:     Test-2-Flag ← 0.
24: end if
```

This Multi-Scale FL Runner (Algorithm 2) is executed during each FL round via Master-FL (Algorithm 3) at the central server $\mathcal{S}$, we describe Master-FL next. Master-FL tracks model training via Multi-Scale FL Runner (Algorithm 2) at each FL round $t = 1, 2, \cdots$. Additonally, it performs two tests (see line 15, 16 in Algorithm 3) to identify whether a significant drift has occurred. In this regard, **Test 1** intuitively allows the server $\mathcal{S}$ to examine whether a "sudden" drift has occurred in the recent training rounds, by tracking every base algorithm upon its completion via actual loss $F^{(t)}$ and U_t derived from auxiliary loss quantity $\tilde{F}^{(t)}$. Whereas, **Test 2** keeps track of non-stationary drifts that gradually accumulated over longer time windows. In the following, we present more mathematical intuitions behind design of **Test 1,2** via an illustrative example.

A. Unpacking Mathematical Intuitions Behind Test 1, 2

Consider the decomposition of *dynamic regret* expression as follows:

$$R_{[1,t]} = \sum_{\tau=1}^{t} F^{(\tau)}(\mathbf{x}^{(\tau)}) - \sum_{\tau=1}^{t} F^{(\tau)}(\mathbf{x}^{(\tau),*}), \tag{14}$$

$$= \underbrace{\sum_{\tau=1}^{t} \left[F^{(\tau)}(\mathbf{x}^{(\tau)}) - \tilde{F}^{(\tau)} \right]}_{\text{(a)}} + \underbrace{\sum_{\tau=1}^{t} \left[\tilde{F}^{(\tau)} - F^{(\tau)}(\mathbf{x}^{(\tau),*}) \right]}_{\text{(b)}}. \tag{15}$$

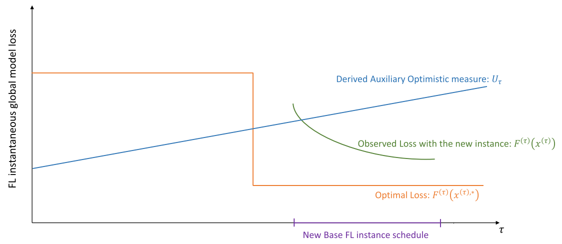


Fig. 2. An illustration of how multiple base FL instances help identifying significant concept drifts.

When *near-stationarity* conditions pertaining to Requirement 1 are met via only a single instance of baseline FL algorithm, then term (a) is $t.\rho(t)$ due to Eq. (13) and term (b) is ≤ 0. The overall regret remains unchanged, i.e., $\tilde{\mathcal{O}}(t.\rho(t))$.

However, beyond the *near-stationarity* regime, i.e., $\Delta_{1,t} > \rho(t)$, both the terms can become drastically large. Note that components of term (a) are observable at the server $\mathcal{S}$ via periodic synchronizations, thereby allowing easy detection of abnormal changes for term (a). To this end, **Test 2** (line 21-24 in Algorithm 3) takes care of term (a).

Term (b) in Eq. (15) cannot be directly evaluated since optimal models $\mathbf{x}^{(\tau),*}$ are unavailable. Large values of term (b) are owed to the fact that for one or more iterations τ, the instantaneous optimal model $\mathbf{x}^{(\tau),*}$ was possibly sub-optimal at rounds $\leq \tau - 1$, and would have caused global loss to be very high if used in those rounds. It is not possible to detect such models via single instance of one base FL algorithm particularly because one instance naturally explores only optimal gradient directions from its starting model using a fixed learning rate. Our proposed randomized multi-scale orchestration scheme that schedules and maintains several base FL instances particularly to enable detection for term (b). We summarize the idea of how this is achieved via a synthetic example in Figure 2. The derived optimistic quantity in Algorithm 3: $U_t = \max_{\tau \in [t_{new}, t]} \tilde{F}^{(t)}$ follows an increasing curve as depicted in Figure 2. In a stationary environment where $\Delta_{[1,t]} \to 0$, eq. (12) implies that U_τ is always a lower bound of base FL instance's performance for every round τ, which is the reason why we use the term "optimism". Now, if a new base FL instance beats the optimistic sequence U_τ with an observable gap as per Figure 2, this will correspond to a significantly drifting environment. Since, we have term (b) $\leq \sum_{\tau=1}^{t} \left[U_\tau - F^{(\tau)}(\mathbf{x}^{(\tau),*}) \right]$, so tracking this derived quantity sequence $\{U_\tau\}$ and the new base FL instance will intuitively facilitate drift detection and restarting FL learning before term (b) grows abruptly with the stale model. These mathematical intuitions in turn support our design of **Test 1** (line 15-20 in Algorithm 3) that proxies detection of term (b) in Eq. (15).

Here, we emphasize that our designed drift signaling tests significantly deviate from prior drift detection mechanisms in FL literature [4], [9], since the proposed tests are not simply heuristic-based. In fact, the tests are designed via careful decomposition of the *dynamic regret* in a non-stationary environment as suggested by the preceding discussion pertaining to mathematical intuition development for **Test 1, 2**. In Section V, we unfold how the designed signaling mechanisms are tied to worst-case FL *dynamic regret* bounds via comprehensive mathematical analysis. Note that another striking feature of our proposed tests is that they have a unified mathematical formulation and can be wrapped with a variety of baseline FL algorithms, for instance *FedAvg, FedOMD*.

B. Random/Warm Model Initializations at Restart

Finally, the FL learning mechanism begins from scratch triggered via Algorithm 2 if any of the aforementioned tests confirm non-stationarity. We re-iterate that this Master-FL routine is conducted directly at the server $\mathcal{S}$. Also, when we say that "the base FL instances are scheduled to start learning from scratch", it means that the previous learning schedule will now be discarded and replaced by a new learning schedule via Algorithm 1. In mathematical terms, the previous base FL instance collection $\boldsymbol{A}_{prev}$ will be replaced by a new collection $\boldsymbol{A}_{new}$ upon executing Algorithm 1. Also, let $X_{prev} = \{\mathbf{x}^{\mathcal{A}} : \mathcal{A} \in \boldsymbol{A}_{prev}\}$ and $X_{new} = \{\mathbf{x}^{\mathcal{A}} : \mathcal{A} \in \boldsymbol{A}_{new}\}$ denote the ML model sets for the previous and the newly created instance sets respectively (see explicit definition of an instance in line 7 of Algorithm 1). Then, instances in X_{new} can either be "randomly initialized" new model vectors, or they could be populated by "recycling" from X_{prev}. Our theoretical analysis presented next in Section V stay unchanged with "warm/random initializations" upon restarts.

V. Main Results

In this section, we will provide the dynamic regret guarantees of the proposed algorithm. In Section V-A, we derive the *dynamic regret* bound of baseline FL algorithms: *FedAvg*, *FedOMD*, and discuss how the bound is impacted by different degrees of non-stationarity. In Section V-B, we show that multi-scale framework indeed preserves the properties of the base FL instances executed over arbitrary *near stationary* horizons, as well as provide a concrete storage complexity bound for the proposed method.

For ease of our mathematical analysis, here we introduce the notions of "blocks" and "epochs" of consecutive FL training rounds. The idea of a *"block"* is motivated by the observation that in Master-FL (Algorithm 3) training happens only in increasing chunks which are sized 2^k, $k \in \mathbb{N}$ (please refer to line 3-6 of Algorithm 3). More specifically, training restarts inevitably at the end of increasingly sized intervals with lengths $2^0, 2^1, \cdots$. However, if a restart trigger happened at any round for an arbitrary order-k block, then this current block is finished with less than 2^k rounds. Recall that a restart is only triggered via `Test-1` and/or `Test-2` in Algorithm 3. Subsequently, Master-FL (Algorithm 3) restarts FL training from scratch for a new order-$k+1$ block with 2^{k+1} rounds. Next, a mathematical definition of "block" is presented.

Definition 5 (Block): Over the execution horizon of Master-FL (Algorithm 3) for rounds $[1, T]$, $\mathcal{B} = [t_m, E_m]$ is called an order-m block iff $E_m \leq t_m + 2^m - 1$, and training restarts from scratch (via line 3-6 in Algorithm 3) at rounds t_m as well as $E_m + 1 \leq T$.

In the following, we unravel the meaning of an "epoch" which we will utilize afterwards in our theoretical analysis. In contrast to a "block", the notion of an "epoch" is strongly associated to only restart triggers of Master-FL (Algorithm 3). Such restart triggers can possibly be separated across multiple successive blocks or just one single block. In our manuscript, we use "epoch" to denote the collection of all rounds between two consecutive restart triggers. A formal definition for an "epoch" has been outlined next.

Definition 6 (Epoch): Over the execution horizon of Master-FL (Algorithm 3) for rounds $[1, T]$, An interval $\mathcal{E} = [t_0, E]$ consisting of successive FL rounds is an "epoch" if $t_0 = 1$ or a restart was triggered at $t_0 - 1$. Furthermore, $E = T$ or the next restart was triggered at round E.

In Section V-C, we mathematically analyze the regret incurred by a block. Finally, in Section V-D, we characterize the epoch regret and further use it to obtain the mathematical expression for the overall *dynamic regret* incurred by Master-FL (Algorithm 3) over T rounds of Federated Learning.

A. Base FL Algorithm Theoretical Guarantees in Dynamic Environments

In this subsection, we first detail the *dynamic regret* analysis of vanilla *FedAvg* and *FedOMD* algorithms, and interpret how the results are impacted by degree of non-stationarity.

1) FedAvg dynamic regret Analysis: We note that the local `FL-UPDATE`$(\cdot)$ pertaining to *FedAvg* can be written $\forall\, n \in \mathcal{N}$ as:

$$\texttt{FL-UPDATE:}\quad \mathbf{x}_n^{(t)} = \mathbf{x}^{(t-1)} - \eta_t \nabla F_n^{(t)}(\mathbf{x}^{(t-1)}), \tag{16}$$

where η_t is the learning rate/step size during ML training at round t. Consequently, combining with Eq. (44), see the Supplementary Material, gives the global ML model upon aggregation by server $\mathcal{S}$ as:

$$\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)} - \eta_t \sum_{n \in \mathcal{N}} p_n^{(t)} \nabla F_n^{(t)}(\mathbf{x}^{(t-1)}), \tag{17}$$

In the following, we present the mathematical bound for *dynamic regret* of *FedAvg*.

Theorem 1 (Dynamic Regret for Convex Loss Function With FedAvg): Assume that the underlying ML loss measure $f(\cdot;\cdot)$ satisfies Assumption 1 and local learning rates at the DPUs collectively represented by $\mathcal{N}$ are set to $\eta_t = \frac{1}{\sqrt{T}}$ for $t \in \{1, 2, \cdots, T\}$, the cumulative dynamic regret incurred by FedAvg Algorithm is bounded by:

$$R_{[1,T]} \leq \frac{\sqrt{T}}{2}\|\mathbf{x}^{(1)} - \mathbf{x}^*\|^2 + \frac{\mu^2\sqrt{T}}{2} + 2T\Delta_{[1,T]}, \tag{18}$$

Proof: From the definition of *dynamic regret* in (7), we have:

$$R_{[1,T]} = \sum_{t=1}^{T} F^{(t)}(\mathbf{x}^{(t)}) - \sum_{t=1}^{T} F^{(t)}(\mathbf{x}^{(t),*}), \tag{19}$$

$$= \underbrace{\sum_{t=1}^{T} F^{(t)}(\mathbf{x}^{(t)}) - \sum_{t=1}^{T} F^{(t)}(\mathbf{x}^*)}_{\text{(a)}}$$

$$+ \underbrace{\sum_{t=1}^{T} F^{(t)}(\mathbf{x}^*) - \sum_{t=1}^{T} F^{(t)}(\mathbf{x}^{(t),*})}_{\text{(b)}}, \tag{20}$$

where, we define the *static comparator* $\mathbf{x}^*$ as follows:

$$\mathbf{x}^* = \min_{\mathbf{x}} \sum_{t=1}^{T} F^{(t)}(\mathbf{x}). \tag{21}$$

Next, we individually bounding terms (a) and (b) in Eq. (46), see the Supplementary Material, and the detailed steps are given in Appendix C, see the Supplementary Material. ■

2) FedOMD Dynamic Regret Analysis: In the following, we first explain the FL-UPDATE$(\cdot)$ rule at the DPUs in $\mathcal{N}$ for *FedOMD* algorithm. In this context, we summarize the mathematical details around the notion of Bregman Divergence. Specifically, consider an arbitrary 1-strongly convex function $\phi : \mathbb{R}^d \to \mathbb{R}$ w.r.t. L2 euclidean norm i.e., $\|\cdot\|$. Therefore, due to strong convexity of $\phi(\cdot)$, the following holds:

$$\phi(\mathbf{y}) \geq \phi(\mathbf{x}) + \langle \mathbf{y}-\mathbf{x}, \nabla\phi(\mathbf{x})\rangle + \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|^2, \ \forall \mathbf{x},\mathbf{y} \in \mathbb{R}^d. \tag{22}$$

Consequently, the Bregman Divergence w.r.t $\phi(\cdot)$ is defined as:

$$B_\phi(\mathbf{y};\mathbf{x}) \triangleq \phi(\mathbf{y}) - \phi(\mathbf{x}) - \langle \mathbf{y}-\mathbf{x}, \nabla\phi(\mathbf{x})\rangle. \tag{23}$$

Furthermore, the Bregman Divergence B_ϕ is chosen such that it satisfies the assumption stated next.

Assumption 2: For any collection of arbitrary points $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_m \in \mathbb{R}^d$, with scalar weights $w_1, \cdots, w_j \in [0,1]$ such that $\sum_{i=1}^{j} w_i = 1$, the following holds for all $u \in \mathbb{R}^d$:

$$B_\phi(u, \sum_{i=1}^{j} w_i z_i) \leq \sum_{i=1}^{j} w_i B_\phi(u, z_i). \tag{24}$$

It is worth highlighting that Assumption 2 is in fact true for commonly used Bregman divergences such as Euclidean Distance and KL-Divergence. Furthermore, we note that this assumption is only required to achieve sub-linear convergence speeds for *FedOMD* algorithm [15], and not a general requirement for our multi-scale algorithmic framework.

The local model update, i.e., FL-UPDATE$(\cdot)$ during each Federated Learning round where *FedOMD* Algorithm is executed can be described as:

$$\mathbf{x}_n^{(t+1)} = \underset{x \in \mathbb{R}^p}{\arg\min}\ \psi_n(\mathbf{x};\mathbf{x}^{(t)}), \tag{25}$$

$$\psi_n(\mathbf{x};\mathbf{x}^{(t)}) \triangleq \langle \nabla F_n^{(t)}(\mathbf{x}^{(t)}), \mathbf{x}\rangle + \frac{1}{\eta_t} B_\phi(\mathbf{x};\mathbf{x}^{(t)}). \tag{26}$$

We restate the global ML model aggregation procedure as summarized in Eq. (44), see the Supplementary Material, in the following:

$$\mathbf{x}^{(t+1)} = \sum_{n \in \mathcal{N}} p_n^{(t)} \mathbf{x}_n^{(t+1)}. \tag{27}$$

Theorem 2 (Dynamic Regret With Convex Loss Function for FedOMD): Assume that the underlying ML loss measure $f(\cdot;\cdot)$ satisfies Assumption 1 and local learning rates at the DPUs collectively represented by $\mathcal{N}$ are set to $\eta_t = \frac{1}{\sqrt{T}}$ for $t \in \{1, 2, \cdots, T\}$, the cumulative dynamic regret incurred by FedOMD Algorithm is bounded by:

$$R_{[1,T]} \leq \sqrt{T} B_\phi(\mathbf{x}^*, \mathbf{x}^{(1)}) + \frac{\mu^2}{2}\sqrt{T} + 2T\Delta_{[1,T]}. \tag{28}$$

Proof: Please refer to Appendix D, see the Supplementary Material. ■

In a nutshell, proving Theorem 2 uses the same *dynamic regret* decomposition technique as done in Theorem 1 via Eq. (45) - (46), see the Supplementary Material. Consequently, the static and dynamic components are individually bounded and combined to get the final *dynamic regret* bound for *FedOMD*.

3) Interpretation of Regret Results Over Varying Degree of Drifts: The *dynamic regret* results obtained in Theorems 1 and 2 corroborate a crucial intuition that supports our algorithm design. More specifically, it reflects that the aforementioned conventional FL methods may work well in dynamic environments which are *near stationary*. To see this mathematically, we appeal to the notion of *near stationarity* introduced in Assumption 1 which implies : $\Delta_{[1,T]} \leq \rho(T)$.

Remark 3: In development and analysis of our algorithmic framework, we have considered full-batch gradient computations for base FL algorithms FedAvg, FedOMD. However, it is worth highlighting that in a mini-batch stochastic gradient computation setting, the statistical properties of the proxy gradient resembles that of the true gradient. Consequently, our algorithmic framework is directly extendable to a stochastic update approach, we present a proof sketch with mathematical details and explanations in Appendix K, see the Supplementary Material.

Remark 4: Although the performance guarantees of the vanilla algorithms don't change at such small drifts, however higher drifts cause the bounds to get worse (from $\tilde{\mathcal{O}}(\sqrt{T})$ in small drift scenarios to $\tilde{\Omega}(\sqrt{T})$ in presence of high drifts) due to linear factoring of the drift terms with horizon length T.

In order to mathematically delineate the claim of Remark 4, we first note that the terms $\Delta_{[1,T]}T$ in the RHS of regret expressions presented via Theorems 1, 2 reflect this aforementioned *linear factoring* effect. To appreciate how exactly theoretical performance of vanilla algorithms deteriorates in high drift scenarios, consider the violation of the *near-stationarity* condition stated in Assumption 1 i.e., $\Delta_{[1,T]} > \rho(t) \geq \frac{1}{\sqrt{T}}$. Consequently, the bounds of vanilla *FedAvg*, *FedOMD* would become at least $\tilde{\Omega}(\sqrt{T})$ in contrast to the *near-stationarity* scenario where the bounds would be at worst $\tilde{\mathcal{O}}(\sqrt{T})$, as suggested by RHS of regret bounds presented in Theorems 1, 2.

To mitigate this, our multi-scale algorithmic framework leverages the *near stationarity* guarantees of the base algorithm by augmenting it with carefully curated non-stationarity tests (**Test 1** and **Test 2** in Algorithm 3). We outline our theoretical findings of how better regrets could be achieved at higher degrees of drifts in the subsequent discussions.

B. Multi-Scale Algorithm Analysis

The Multi-Scale FL Runner (Algorithm 2) is essentially a subroutine within Master-FL (Algorithm 3) that orchestrates different scheduled base FL instances over a specified block of aggregation rounds, and produces the sequence of optimistic loss quantities $\{\tilde{F}^{(t)}\}$. We redirect the readers to Appendix E, see the Supplementary Material, wherein we outline the construction of the sequence $\{\tilde{F}^{(t)}\}$ and validate Requirement 1 for *FedAvg, FedOMD*.

Lemma 1: Let $\hat{m} = \log_2 T + 1$ and Multi-Scale FL Runner (Algorithm 2) is executed with input $m \le \log_2 T$. Furthermore, we assume that with any instance of base FL algorithm $\mathcal{A}$ initiated within Algorithm 2 and any $t \in [\mathcal{A}.s, \mathcal{A}.e]$, the cumulative concept drift satisfies $\Delta_{[\mathcal{A}.s,t]} \le \rho(t')$ where $t' = t - \mathcal{A}.s + 1$. Then, with probability $1 - \frac{\delta}{T}$, the following holds:

$$\tilde{F}^{(t)} \le \max_{\tau \in [\mathcal{A}.s,t]} F^{(\tau)}(\mathbf{x}^{(\tau),*}) + \Delta_{[\mathcal{A}.s,t]}, \tag{29}$$

$$\frac{1}{t'} \sum_{\tau=\mathcal{A}.s}^{t} F^{(\tau)}(\mathbf{x}^{(\tau)}) - \tilde{F}^{(\tau)} \le \hat{\rho}(t') + \hat{m}\Delta_{[\mathcal{A}.s,t]}, \tag{30}$$

and, the number of instances running within the interval $[\mathcal{A}.s, t]$ is bounded by $6\hat{m}\log(T/\delta)\frac{C(t')}{C(1)}$, where $C(t)$ is as described in Definition 3.

Proof: Please refer to Appendix G, see the Supplementary Material. ■

Interpretation of Lemma 1 Results: The first part of the Lemma ensures that the behavior of the baseline FL methods, i.e., *FedAvg and FedOMD* remain unchanged even under the proposed multi-scale orchestration scheme. Formally, it proves a more general version of the requirements specified in Assumption 1 for arbitrary time intervals, under the assumption that they are *near-stationary*. Furthermore, it provides a storage complexity analysis in terms of worst case bound on the number of base FL instances scheduled as $\tilde{\mathcal{O}}(C(T))$ over a horizon of T FL rounds.

C. Block Regret Analysis

In this subsection, we summarize the *dynamic regret* for any such arbitrary block of order m (see Definition 5). More specifically, we consider this block scheduled to run for the rounds $[t_m, t_m + 2^m - 1]$. In proving the block *dynamic regret* result of Lemma 2 stated later, we basically divide $[t_m, t_m + 2^m - 1]$ into successive intervals $\mathcal{I}_1 = [s_1, e_1]$, $\mathcal{I}_2 = [s_2, e_2]$, $\cdots$, $\mathcal{I}_K = [s_K, e_K]$ ($s_1 = t_m, e_i + 1 = s_{i+1}, e_K = t_m + 2^m - 1$). Furthermore, all these intervals are assumed to satisfy:

$$\Delta_{\mathcal{I}_i} \le \rho(|\mathcal{I}_i|), \ \forall\ i. \tag{31}$$

For our current set of baseline algorithms: *FedAvg* and *FedOMD*, according to Theorem 1, 2 we can have $C(t) = t\rho(t) = \min\{c_1\sqrt{t} + c_2, t\}$ since the losses are bounded in $[0, 1]$.

Lemma 2 (Block Dynamic Regret): Let $\mathcal{B} = [t_m, E_m]$ be an order-m block (see Definition 5) for which Master-FL (Algorithm 3) is executed. Then, the dynamic regret incurred on $\mathcal{B}$ is bounded as:

$$R_{\mathcal{B}} \le \tilde{\mathcal{O}}\Big(\min\Big\{R_L(\mathcal{B}), R_\Delta(\mathcal{B})\Big\} + \Big(c_1 + \frac{c_2}{c_1}\Big)2^{m/2} + c_2^2\Big), \tag{32}$$

where, $R_L(\mathcal{B}), R_\Delta(\mathcal{B})$ are defined as follows:

$$R_L(\mathcal{B}) \triangleq c_1\sqrt{L|\mathcal{B}|} + c_2 L, \tag{33}$$

$$\begin{aligned} R_\Delta(\mathcal{B}) \triangleq\ & c_1^{\frac{2}{3}}\Delta^{\frac{1}{3}}|\mathcal{B}|^{\frac{2}{3}} + c_1\sqrt{|\mathcal{B}|} + c_1\sqrt{\Delta|\mathcal{B}|} \\ & + c_2 c_1^{-\frac{2}{3}}\Delta^{\frac{2}{3}}T^{\frac{1}{3}} + c_2(1+\Delta), \end{aligned} \tag{34}$$

and, where L, Δ are as described in Definition 1, 2.

Proof: Please refer to Appendix H, see the Supplementary Material. ■

Interpretation of Lemma 2 Results: The results presented in this lemma offer a key insight regarding how Master-FL behavior on each small block of FL rounds. Roughly speaking, it leverages the fact that every arbitrary block of order m contains smaller intervals which are *near-stationary*. Then, it achieves the stated result by combining the result in Lemma 1 and an upper bound on how many such smaller *near-stationary* segments may exist in a given block. Furthermore, it expresses the regret as the smaller of the quantities $R_L(\mathcal{B}) = \tilde{\mathcal{O}}(\sqrt{L|\mathcal{B}|} + L)$ and $R_\Delta(\mathcal{B}) = \tilde{\mathcal{O}}(\Delta^{\frac{1}{3}}|\mathcal{B}|^{\frac{2}{3}} + \sqrt{|\mathcal{B}|})$ for any arbitrary $\mathcal{B}$. In Section V-D, we show how this result translates to a concrete mathematical bound for the overall *dynamic regret* of Master-FL.

D. Dynamic Regret Analysis of Master-FL

In this section, we first present the *dynamic regret* result pertaining to a single epoch of FL rounds for which Master-FL (Algorithm 3) conducts training. We reiterate that an epoch is essentially an interval between two consecutive restart triggers.

Lemma 3 (Single Epoch Regret Analysis): Let, $\mathcal{E} = [t_0, E]$ be an epoch containing blocks upto order m for which Master-FL (Algorithm 3) runs. Then, the dynamic regret associated with $\mathcal{E}$ is:

$$R_{\mathcal{E}} \le \tilde{\mathcal{O}}\Big(\min\Big\{R_L(\mathcal{E}), R_\Delta(\mathcal{E})\Big\} + (c_1 + \frac{c_2}{c_1})\sqrt{|\mathcal{E}|} + c_2^2\Big). \tag{35}$$

where $R_L(\cdot)$, $R_\Delta(\cdot)$ are as defined in Lemma 2.

Proof: Please refer to Appendix I-A, see the Supplementary Material. ■

Lemma 4 (Statistical Consistency of Stationary Test Trigger Events): Let t be a FL round within an epoch starting from t_0. If $\Delta_{[t_0,t]} \le \rho(t - t_0 + 1)$, then with high probability, no restart is triggered during round t.

Proof: Please refer to Appendix I-B, see the Supplementary Material. ■

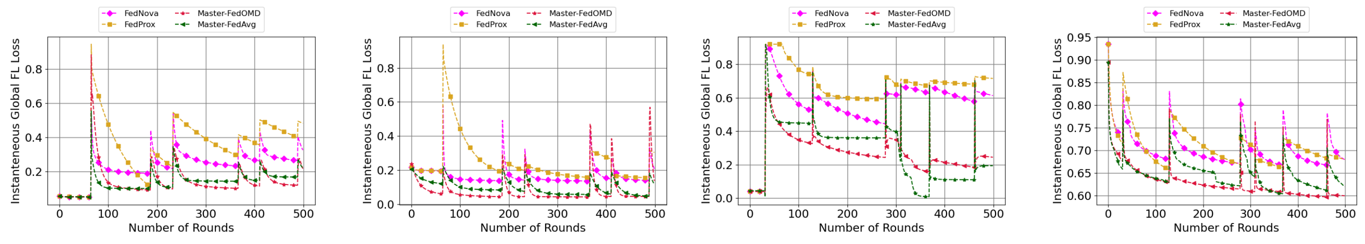
Lemma 5 (Bound for Number of Epochs M Over the Entire Horizon): For total number of epochs denoted by M over the horizon of length T on which Master-FL is executed, the following results hold with high probability:

$$M \le L, \tag{36}$$

$$M \le 1 + 2c_1^{-\frac{2}{3}}\Delta^{\frac{2}{3}}T^{\frac{1}{3}} + \Delta. \tag{37}$$

Proof: Please refer to Appendix I-C, see the Supplementary Material. ■

Theorem 3 (Dynamic Regret of Master-FL Algorithm): Assume that individual base FL instances satisfy the conditions specified in Assumption 1, then without the knowledge of non-stationary measures L or Δ, Master-FL (Algorithm 3) incurs dynamic regret over T FL aggregation rounds which



(a): Covtype (CI drift). (b): Covtype (CS drift). (c): MNIST (CI drift). (d): MNIST (CS drift).

Fig. 3. Instantaneous global FL loss vs aggregation rounds for various methods. CS $\longrightarrow$ Class Swap Drift, CI $\longrightarrow$ Class Introduction Drift.

is bounded by:

$$\mathcal{R}_{[1,T]} = \tilde{\mathcal{O}}\Big(\min\Big\{\Big(c_1 + \frac{c_2}{c_1}\Big)\sqrt{LT}, c_2 c_1^{-\frac{4}{3}}\Delta^{\frac{1}{3}}T^{\frac{2}{3}} + \sqrt{T}\Big\}\Big). \tag{38}$$

with high probability.

Proof: Please refer to Appendix J, see the Supplementary Material. ■

Interpretation of Lemma 3 - 5, Theorem 3 Results: At a high level, Lemma 3 verifies that the block *dynamic regret* result directly translates to an epoch with multiple blocks as long as the change-detection/restart events are correctly triggered via **Test 1**, **Test 2** in Master-FL. Therefore, in the next step, we formally verify that restarts are not triggered as long as the blocks of FL rounds are *near-stationary* in Lemma 4. With the exact mathematical bound in place for regret incurred in each epoch and the proof of correctness of restart triggers, we move to characterizing the upper limit on number of epochs in terms of horizon length and non-stationary measures, i.e., $T, L,$ and Δ respectively in Lemma 5. Finally, we combine the bounds of each individual epoch *dynamic regret* and number of epochs, to obtain the final regret bound incurred by Master-FL in Theorem 3.

VI. Experimental Evaluations

In this section, we perform proof-of-concept experiments comparing Master-FL-*FedAvg* and Master-FL-*FedOMD* with the closest competing algorithms *FedNova* [20] and vanilla *FedProx* [23] on 2 LIBSVM [1] classification datasets: covtype, mnist. *FedNova* and *FedProx* are widely-used benchmark FL methods that mitigate drifts collected in the global objective via careful modifications in their optimization frameworks. The aforementioned datasets are extensively leveraged for experimental studies pertaining to FL and distributed ML architectures [24], [25], as well as can be conveniently used to simulate time-varying non-stationary scenarios.

For our experiments, we consider L1 regularized multi-class logistic regression formulation for the underlying ML loss function, i.e., $f(\cdot;\xi)$ for datapoint ξ. More precisely, for a model $\mathbf{x}$ and datapoint ξ, we have:

$$f(\mathbf{x};\xi) = \log\big(1 + exp(-\xi_{(2)}.\mathbf{x}^T\xi_{(1)})\big) + \frac{\lambda}{2}\|x\|_2, \tag{39}$$

[1]LIBSVM Collection of dataset can be accessed at https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

TABLE II

Average FL Classification Accuracy for Various Methods Across Total $T = 500$ Rounds. CS $\longrightarrow$ Class Swap Drift, CI $\longrightarrow$ Class Introduction Drift

	covtype		**mnist**	
	CI	CS	CI	CS
Master-FL-*FedAvg*	0.874	0.852	0.719	0.739
Master-FL-*FedOMD*	0.876	0.864	0.682	0.780
FedNova	0.799	0.753	0.629	0.644
FedProx	0.673	0.686	0.577	0.601

wherein every datapoint is defined as $\xi \triangleq (\xi_{(1)}, \xi_{(2)})$, i.e., tuple of the feature vector $\xi_{(1)}$ and class label $\xi_{(2)}$ respectively. We use grid-search for tuning regularization parameter[2] and presented results are for $\lambda = 2e-4$. Additionally, the learning rates for the different algorithms, η, is chosen as $\eta = 1/\sqrt{T}$ for T rounds. The number of client DPUs are $|\mathcal{N}| = 20$, and we conduct FL training upto $T = 500$ rounds. During every FL training round t, each DPU $n \in \mathcal{N}$ acquires its local dataset $\mathcal{D}_n^{(t)}$ with sizes distributed as $\mathcal{N}(1000, 200)$ via random sampling without replacement from the core data-source.

We consider two different categories of *concept drift* that are particularly aligned with our classification tasks: *class introduction (CI) drift* and *class swap (CS) drift* [25]. In CI experiments, new classes are injected into the DPUs at certain rounds, thereby adding a non-stationary shock to the system. For "covtype" dataset, we add CI drift for the rounds $t \in \{65, 187, 233, 367, 411, 489\}$ wherein the classes are added sequentially in the order $\{0\} \rightarrow \{1\} \rightarrow \{2\} \rightarrow \{3\} \rightarrow \{4\} \rightarrow \{5\} \rightarrow \{6\}$ starting from $t = 0$ and at each drift round in the set. On the other hand, for "mnist" dataset, CI drift is injected at rounds $t \in \{31, 129, 279, 310, 369, 462\}$ wherein the classes are added in the order $\{0, 1\} \rightarrow \{2, 3\} \rightarrow \{4\} \rightarrow \{5, 6\} \rightarrow \{7\} \rightarrow \{8\} \rightarrow \{9\}$. In CS experiments, labels are swapped for pair(s) of classes. We follow the same schedule for injecting shock into the system. At each drift round, we used 3 pairs of classes for swapping, i.e., $\{(0, 1), (2, 3), (4, 5)\}$.

Figure 3 demonstrates how the performance of the FL methods are impacted by drift shocks. Observe that global losses spike approximately around the drift injection rounds, signifying unexpected non-stationarity experienced by the

[2]Note that *FedNova* and *FedProx* use L2 regularization instead of L1 in Eq. (39).

network of DPUs. Consequently, note that Master-FL-*FedAvg*, Master-FL-*FedOMD* outperform competing FL methods for all the dataset-drift combinations in terms of instantaneous FL loss, and this is also corroborated by the cumulative average FL classification accuracies presented in Table II. The results obtained supports our framework's ability to quickly re-train foregoing models that were impacted by *concept drift*.

VII. Conclusion

In this paper, we propose Master-FL a multi-scale change detection-restart based algorithmic framework that can leverage any baseline FL optimizer that works well in *near-stationary* environments to adaptively learn in a highly drifting environment. We derive online *dynamic regret* bounds for vanilla *FedAvg* and *FedOMD*, as well as demonstrate their *near-stationary* properties which are essential for augmentation with Master-FL. Subsequently, we provide rigorous mathematical analysis for convex loss functions leading to *dynamic regret* bounds in terms of non-stationary measures Δ, L with no prior knowledge requirement or stronger convexity assumptions. To the best of our knowledge, the *dynamic regret* bounds revealed in this work are novel in the non-stationary federated optimization setting, while it extends the existing literature in the general centralized as well as distributed online optimization paradigms. As a future direction, extending to other FL baseline algorithms which may demonstrate favorable performance under stationary assumptions definitely deserves consideration. Further, considering client selection approaches [26] for heterogeneous non-stationary data is an open problem.

References

[1] S. Hosseinalipour, C. G. Brinton, V. Aggarwal, H. Dai, and M. Chiang, "From federated to fog learning: Distributed machine learning over heterogeneous wireless networks," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 41–47, Dec. 2020.

[2] A. Garg, L. Marla, and S. Somanchi, "Distribution shift in airline customer behavior during COVID-19," 2021, *arXiv:2111.14938*.

[3] B. Ganguly et al., "Multi-edge server-assisted dynamic federated learning with an optimized floating aggregation point," *IEEE/ACM Trans. Netw.*, early access, Apr. 21, 2023, doi: 10.1109/TNET.2023.3262482.

[4] A. Mallick, K. Hsieh, B. Arzani, and G. Joshi, "Matchmaker: Data drift mitigation in machine learning for large-scale systems," in *Proc. Mach. Learn. Syst.*, vol. 4, 2022, pp. 77–94.

[5] M. Sugiyama and M. Kawanabe, *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation*. Cambridge, MA, USA: MIT Press, 2012.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[7] L. Yang and A. Shami, "A lightweight concept drift detection and adaptation framework for IoT data streams," *IEEE Internet Things Mag.*, vol. 4, no. 2, pp. 96–101, Jun. 2021.

[8] A. Abbasi, A. R. Javed, C. Chakraborty, J. Nebhen, W. Zehra, and Z. Jalil, "ElStream: An ensemble learning approach for concept drift detection in dynamic social big data stream learning," *IEEE Access*, vol. 9, pp. 66408–66419, 2021.

[9] F. E. Casado, D. Lema, M. F. Criado, R. Iglesias, C. V. Regueiro, and S. Barro, "Concept drift detection and adaptation for federated and continual learning," *Multimedia Tools Appl.*, vol. 81, no. 3, pp. 3397–3419, Jan. 2022.

[10] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 928–936.

[11] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *18th Int. Conf. Artif. Intell. Statist. (PMLR)*, vol. 38, 2015, pp. 398–406.

[12] S. Shahrampour and A. Jadbabaie, "Distributed online optimization in dynamic environments using mirror descent," *IEEE Trans. Autom. Control*, vol. 63, no. 3, pp. 714–725, Mar. 2018.

[13] K. Lu, G. Jing, and L. Wang, "Online distributed optimization with strongly pseudoconvex-sum cost functions," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 426–433, Jan. 2020.

[14] X. Li, X. Yi, and L. Xie, "Distributed online convex optimization with an aggregative variable," *IEEE Trans. Control Netw. Syst.*, vol. 9, no. 1, pp. 438–449, Mar. 2022.

[15] A. Mitra, H. Hassani, and G. J. Pappas, "Online federated learning," in *Proc. 60th IEEE Conf. Decis. Control (CDC)*, Oct. 2021, pp. 4083–4090.

[16] S. Hosseinalipour et al., "Parallel successive learning for dynamic distributed model training over heterogeneous wireless networks," *IEEE/ACM Trans. Netw.*, early access, Jul. 10, 2023, doi: 10.1109/TNET.2023.3286987.

[17] R. Dixit, A. S. Bedi, and K. Rajawat, "Online learning over dynamic graphs via distributed proximal gradient algorithm," *IEEE Trans. Autom. Control*, vol. 66, no. 11, pp. 5065–5079, Nov. 2021.

[18] N. Eshraghi and B. Liang, "Improving dynamic regret in distributed online mirror descent using primal and dual information," in *Proc. Learn. Dyn. Control Conf.*, 2022, pp. 637–649.

[19] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-iid features via local batch normalization," in *Proc. Int. Conf. Learn. Represent.*, 2021. [Online]. Available: https://openreview.net/forum?id=6YEQUn0QICG

[20] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, Dec. 2020, pp. 7611–7623.

[21] C.-Y. Wei and H. Luo, "Non-stationary reinforcement learning without prior knowledge: An optimal black-box approach," in *Proc. Conf. Learn. Theory*, 2021, pp. 4300–4354.

[22] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–36, Jul. 2022.

[23] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[24] D. Kovalev, E. Gasanov, A. Gasnikov, and P. Richtarik, "Lower bounds and optimal algorithms for smooth and strongly convex decentralized optimization over time-varying networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 22325–22335.

[25] G. Canonaco, A. Bergamasco, A. Mongelluzzo, and M. Roveri, "Adaptive federated learning in presence of concept drift," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2021, pp. 1–7.

[26] F. Fourati, S. Kharrat, V. Aggarwal, M.-S. Alouini, and M. Canini, "FilFL: Client filtering for optimized client participation in federated learning," 2023, *arXiv:2302.06599*.

[27] T. Ando, "Matrix young inequalities," in *Operator Theory in Function Spaces and Banach Lattices*. Cham, Switzerland: Springer, 1995, pp. 33–38.

[28] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "On the generalization ability of on-line learning algorithms," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 2050–2057, Sep. 2004.

[29] K. Azuma, "Weighted sums of certain dependent random variables," *Tohoku Math. J.*, vol. 19, no. 3, pp. 357–367, Jan. 1967.

[30] S. Bernstein, *The Theory of Probabilities*. Gostechizdat, Moscow: Gastehizdat Publishing House, 1946.