# FLHetBench: Benchmarking Device and State Heterogeneity in Federated Learning

## Supplementary Material

## I. More Details about Metrics, Sampling Methods and Proof

### I.1. More Details about Metrics

#### I.1.1 FL Communication Cost $t_{cost}$

A client's communication cost $t_{cost}$ for one round can be approximated by accounting for three factors: the ideal computation and communication costs dependent on device capacity, an extended cost due to state unavailability, and additional costs such as server configuration and I/O delay. This can be represented by the following equation:

$$t_{cost} = \frac{|\theta|}{B_{up}} + \frac{|\theta|}{B_{down}} + |D|\, S_{comp} + t_{unavail} + t_{break}. \quad (3)$$

The first three components, $\frac{|\theta|}{B_{up}} + \frac{|\theta|}{B_{down}} + |D|\, S_{comp}$, represent the ideal computation and communication costs, assuming clients are always available. Here, $|\theta|$ represents the communicated model size, $B_{up}$ and $B_{down}$ indicate network uploading and downloading speeds, $|D|$ refers to the size of a client's local training data, and $|S_{comp}|$ denotes the training speed. The term $t_{unavail}$ represents the client's unavailable duration during the completion of the uploading, downloading, and training process. Specifically, Let $Stat_i$ represent the state trace of client $i$, where $Stat_i[t_1 : t_2]$ indicates the available time duration of client $i$ from time $t_1$ to $t_2$. So, the $t^i_{cost}$ of client $i$ at time $t$ needs to satisfy $Stat_i\left[t : t + t^i_{cost}\right] = t^i_{cost} - t_{unavail}$. Additional costs, such as server configuration and I/O delay, are captured by $t_{break}$.

In our MC simulated FL, we calculate $t_{cost}$ by assuming that $|\theta|$ is the same model size as ViT-B and $|D|$ to be 1. The values of $B_{up}$, $B_{down}$, and $S_{comp}$ can be inferred from our database of device capabilities. We set $t_{break}$ to a constant of 20 seconds.

#### I.1.2 Metrics for Deadline-based Strategies

**Calculation of $\mathcal{S}_i$ for DevMC-R:** To measure sole device heterogeneity, we assume clients' states are always available, which means $t_{unavail} = 0$ in Eq. (3), and propose to use MC motived Algorithm S3 to measure $\mathcal{S}_i$. In each MC simulated FL communication round, we randomly sample $k$ clients and calculate the corresponding $t^i_{cost}$ of client $i$ based on their associated device capability $Dev_i$ and Eq. (3). We then check whether $t^i_{cost}$ is less than the specified $ddl$ and record their successful times by $\mathcal{S}_i$.

**Calculation of $\mathcal{S}_i$ for StatMC-R and InterMC-R:** We use Algorithm S4 (with only black texts) and Algorithm S4 (with both black and red texts) to derive $\mathcal{S}_i$ for StatMC-R and InterMC-R, respectively. The only difference between StatMC-R and InterMC-R is the calculation of $t^i_{cost}$. Specifically, to assess state heterogeneity, we assume $t^i_{cost}$ follow a prior uniform distribution $p\left(t^i_{cost}\right)$ of $(0, ddl)$. However, we use the real device databases $Dev$ to estimate the $t^i_{cost}$ for InterMC-R.

For accurately calculating $\mathcal{S}_i$ for DevMC-R, StatMC-R, and InterMC-R, we establish a total number of simulated FL communication rounds, denoted as $rounds$. It is advisable to set the value of $rounds$ large enough to encompass the entire time duration of $Stat$. In our empirical analysis, we set $rounds$ to 3000 to ensure reliable results.

**From $\mathcal{S}_i$ to metrics for deadline-based Strategies.** We introduce two decay functions ($F(x) = \log(x+1)$ and $Clip\left(\mathcal{S}_i, 0, \mathcal{S}_{ideal}\right)$) that account for the diminishing contribution of increases in total number of successful participation times, see in Eq. (1). $Clip$ is defined as:

$$Clip\left(\mathcal{S}_i, 0, \mathcal{S}_{ideal}\right) = \begin{cases} 0, & \mathcal{S}_i \leq 0 \\ \mathcal{S}_i, & 0 < \mathcal{S}_i < \mathcal{S}_{ideal} \\ \mathcal{S}_{ideal}, & \mathcal{S}_i \geq \mathcal{S}_{ideal}. \end{cases}$$

Here $\mathcal{S}_{ideal}$ is determined by assuming all clients are always available, and all device capacities are the same: $\mathcal{S}_{ideal} = \frac{k}{N} * rounds$, where $rounds$ and $k$ are the same as in calculation of $\mathcal{S}_i$. The $Clip$ function serves as an upper bound for more active clients resulting from device and state heterogeneity, as it limits the influence of excessive successful participation times on the overall metric. This is essential because excessive participation from highly active clients can potentially skew the evaluation and mask the impact of other clients in the system. Similarly, $F(x) = \log(x + 1)$ is based on the intuition that as the number of participation times increases, a client's contribution to the global model should decrease, promoting fairness and diversity among clients.

#### I.1.3 Metrics for Readiness-based Strategies.

In practical FL training using readiness-based strategy, researchers usually allow FL training to continue until a target accuracy of server model is reached and record the total communication cost (or total training time) for evaluation. This approach is different from the deadline-based strategy, which relies on fixed communication rounds $rounds$. However, the readiness-based strategy can lead to unpredictable final communication rounds, as device and state heterogeneity can intervene, causing variations in the number of active clients participating in different rounds. For example, there might be fewer participating clients in each round due to device and state heterogeneity, which means the server model will require more communication rounds to achieve the target accuracy. Therefore, using $rounds$ as the endpoint for our metrics' Monte Carlo calculation is not appropriate. Instead, we opt for the concept of $trips$, referring to the number of client updates received at the server upon training completion, which is widely used in Asynchronous FL [21, 36].

**Total communication cost $T$ estimation for DevMC-T:** In the readiness-based FL aggregation strategy, the server waits for a specified proportion, denoted as $pr$, of clients to complete their tasks, without imposing a deadline (ddl). Algorithm S1 displays the process of evaluating the total communication cost for DevMC-T, with state heterogeneity disabled by setting all clients' states to be always available and $t_{unavail} = 0$. For each MC simulated FL

communication round, we randomly sample $k$ clients and obtain their corresponding $t^i_{cost}$ based on $Dev_i$ and Eq. (3). The communication cost of the current FL round, denoted as $t^{pr}_{cost}$, is then updated as $t^{pr}_{cost} \leftarrow Q_{pr}(t^i_{cost}|i \in round\_ids)$, where $Q_{pr}$ represents the $pr$-th quantile. This $Q_{pr}$ aids in simulating the real readiness-based FL aggregation strategy, where the server waits for a proportion $pr$ of clients to complete their tasks.

**Total communication cost $T$ estimation for StatMC-T and InterMC-T:** The black text in Algorithm S2 depicts how we assess $T$ with only state heterogeneity. Similar to Algorithm S4, we check the available states of clients based on $Stat$ when selecting clients and randomly sample clients from $ready\_ids$. We also assume $t^i_{cost}$ follow a prior uniform distribution $p\left(t^i_{cost}\right)$ of $(0, ddl)$ and update it with $t_{unavail}$ based on $Stat_i$. Finally, using the quantile function $Q_{pr}(t^i_{cost}|i \in round\_ids)$ to get the communication cost for updating $T$. Noting that due to inavailability of clients accouting for state heterogeneity, $cur\_trips$ only adds the number of selected clients $ready\_ids$. The black text and red text in Algorithm S2 formulate the estimation for InterMC-T. The primary distinction between InterMC-T and StatMC-T, denoted by the red text, lies in the fact that the $t^i_{cost}$ for InterMC-T is derived from $Dev_i$, taking device heterogeneity into consideration.

For readiness-based strategy, we repeat above steps and update $cur\_trips$ accordingly until $cur\_trips$ reaching aimed $trips$.

**From $T$ to Metrics for Readiness-based strategy.** We use a constant denominator $\frac{trips}{k} \times t_{break}$ to scale down the metric's magnitude, as $T$ is usually in the millions of seconds, making it easier to interpret and compare. Specifically, the metrics for readiness-based strategy are updated to:

$$\textbf{DevMC-T}, \textbf{StatMC-T}, \textbf{InterMC-T} = \frac{T}{\frac{trips}{k} \times t_{break}}$$

where $t_{break}$ is empirically set to 20 seconds as in Eq. (3), $k$ is the number of sampled clients per round, and $trips$ is set to 10000.

**Algorithm S1** Communication cost estimation for device metric **DevMC-T** under client readiness-based FL aggregation strategy with a specified $pr$ proportion of clients to complete their task during each FL round.

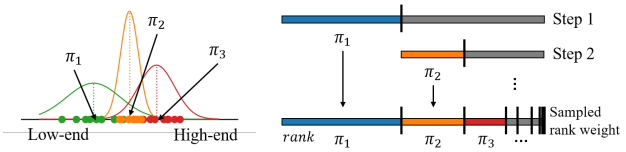
**Require:** Total FL training trips $trips$, Total $N$ local clients with their associated device capacities $Dev$ with upload/computation/download speeds, $k$ the number of clients selected in each round, $t_{break}$ the cost for server allocation and aggregation, $Q_{pr}$ represents the $pr$ quantile function.

Initialize $T \leftarrow 0, cur\_trips \leftarrow 0$
**while** $cur\_trips < trips$ **do**
  $round\_ids \leftarrow$ Sample $k$ clients
  $cur\_trips \leftarrow cur\_trips + k$
  **for** client $i$ in $round\_ids$ **do**
    Extract $ids$ device capacity $Dev_i$ from $Dev$
    Use $Dev_i$ to get its cost $t^i_{cost}$ based on Eq. (3)
  $t^{pr}_{cost} \leftarrow Q_{pr}(t^i_{cost}|i \in round\_ids)$
  $T \leftarrow T + t^{pr}_{cost} + t_{break}$
**return** $T$

**Algorithm S2** Communication cost estimation for state metric **StatMC-T** (or interplay metric InterMC-T) under client readiness-based FL aggregation strategy with a specified $pr$ proportion of clients to complete their task during each FL round.

**Require:** Total FL training client trips $trips$. Total $N$ local clients with their associated states $Stat$ and device capacities $Dev$ with upload/computation/download speeds. $Stat_i[t_1 : t_2]$ the available time duration from time $t_1$ to $t_2$ of client $i$. $k$ the number of clients selected in each round, $t_{break}$ the cost for server allocation and aggregation. $Q_{pr}$ represents the $pr$ quantile function. $t^i_{cost}$ the cost for client $i$ to complete one round of download/computation/upload tasks, which is from uniform distribution and fixed in one simulation.

Initialize $T \leftarrow 0, cur\_trips \leftarrow 0$
**while** $cur\_trips < trips$ **do**
  Find available clients $ready\_ids$ at time $t$ from $Stat$
  **if** $ready\_ids$ is empty **then**
    Wait for another round to sample
  **else if** Number of clients in $ready\_ids < k$ **then**
    Sampled clients $round\_ids \leftarrow ready\_ids$
  **else**
    $round\_ids \leftarrow$ Sample $k$ clients from $ready\_ids$
    $cur\_trips \leftarrow cur\_trips$+Number of clients in $ready\_ids$
  **for** client $ids$ in $round\_ids$ **do**
    (Interplay)
    Extract $ids$ device capacity $Dev_i$ from $Dev$
    Use $Dev_i$ to get its cost $t^i_{cost}$
    Update the $t_{unavail}$ part of Eq. (3) based on $Stat_i$
  $t^{pr}_{cost} \leftarrow Q_{pr}(t^i_{cost}|i \in round\_ids)$
  $T \leftarrow T + t^{pr}_{cost} + t_{break}$
**return** $T$




(a) DPGMM for continous device (b) DPCSM for discrete state

Figure S1. Overview of our DPGMM and DPCSM sampling methods. Low-end/High-end refer to different device capabilities, $\pi_i$ denotes the probability of selecting a particular distribution, and $rank$ represents the rank of the state score.

## I.2. More Details about Sampling Methods and Proof

In Sec. 3.2, we propose DPGMM for sampling continuous device databases and DPCSM for sampling discrete state databases.

For device databases, since the prevalence of the Gaussian distribution in natural phenomena is widely acknowledged, we establish it as the primary distribution for modeling the speed of individual devices. However, due to inherent variations among devices, the speeds of different devices can be grouped into different cate-

**Algorithm S3** Successful participation times $S$ count of device metric **DevMC-R** for deadline-based FL aggregation strategy with report deadline $ddl$.

**Require:** Total FL training rounds $rounds$. Total $N$ local clients with their associated device capacities $Dev$ with upload/computation/download speeds. $k$ the number of clients selected in each round, $t_{break}$ the cost for server allocation and aggregation.

Initialize the success count $\mathcal{S} \leftarrow 0$ and $T \leftarrow 0$
**for** round $= 1$ to $rounds$ **do**
  $round\_ids \leftarrow$ Sample $k$ clients
  **for** client $ids$ in $round\_ids$ **do**
    Extract $ids$ device capacity $Dev_i$ from $Dev$
    Use $Dev_i$ to get its cost $t^i_{cost}$ based on Eq. (3)
    **if** $ddl > t^i_{cost}$ **then**
      $\mathcal{S}^{ids} \leftarrow \mathcal{S}^{ids} + 1$
      $comm\_cost_i \leftarrow t^i_{cost}$
    **else**
      $comm\_cost_i \leftarrow ddl$
  $T \leftarrow T + \max_{i \in round\_ids} comm\_cost_i + t_{break}$
**return** $\mathcal{S}$ of each client

gories, each following its own Gaussian distribution. This leads to a situation where the speed samples within a device database can be considered as originating from a mixture Gaussian distribution. The Dirichlet process is generalized from the Dirichlet distribution and is used to describe the distribution of categories, therefore allowing for flexible modeling and sampling of categories of device speeds. Given a real-world device $\mathcal{D}$, DPGMM receives two parameters, $K_n$ and $\sigma$, to control device heterogeneity. Larger $K_n$ and $\sigma$ shows higher heterogeneity than lower $K_n$ and $\sigma$. Fig. S4 shows an example of the generation of samples for 10 clients with different degrees of device heterogeneity. $K_n$ controls the number of distinct samples; thus, more distinct devices are allocated to these 10 clients when $K_n = 4$ compared to $K_n = 2$. $\sigma$ controls the variation of the selected $K_n$ devices, and when $\sigma$ increases, the variation of the selected devices increases. Therefore, when we increase $K_n$ and $\sigma$, the degree of the device heterogeneity of the samples increases.

For state databases, as a client's state is a collection of discrete data and cannot be characterized by Gaussian distributions like device and is also unsuitable for Dirichlet process. We employ the idea of StatMC-R and use the single state metric (adapted from StatMC-R, see Appendix Sec. V) to transform the discrete state into one single data point, making them compatible with the Dirichlet process. Given a real-world baseline database with $N$ states, DPCSM receives two parameters $startRank$ and $\alpha$ to control heterogeneity. Larger $startRank$ and $\alpha$ shows higher heterogeneity than lower $startRank$ and $\alpha$. Specifically, $startRank$ controls the rank of the optimal state selected from the baseline dataset, thus a lower $startRank$ indicates a higher quality optimal state of the selected $K_n$ states. $\alpha$ controls the probability of selecting subsequent states with lower single state metric, and when $\alpha$ increases, the variation of the selected states increases. Therefore, when we increase $startRank$ and $\alpha$, the degree of the state heterogeneity of the samples increases. Besides, to ensure the sum of probability of selecting each state $\pi_k, k \in 1, ..., N$ equals 1, we increase the probability of last state by $1 - \sum_{k=startRank}^{N} \pi_k$.

**Algorithm S4** Successful participation times $\mathcal{S}$ count of state metric **StatMC-R** (or interplay metric InterMC-R) for deadline-based FL aggregation strategy with report deadline $ddl$.

**Require:** Total FL training rounds $rounds$. Total $N$ local clients with their associated states $Stat$ and device capacities $Dev$ with upload/computation/download speeds. $Stat_i[t_1 : t_2]$ the available time duration from time $t_1$ to $t_2$ of client $i$. $k$ the number of clients selected in each round, $t_{break}$ the cost for server allocation and aggregation. $t^i_{cost}$ the cost for client $i$ to complete one round of download/computation/upload tasks, which is from uniform distribution (0, $ddl$) and fixed in one simulation.

Initialize the success count $\mathcal{S} \leftarrow 0$ and $T \leftarrow 0$
**for** round $= 1$ to $rounds$ **do**
  Find available clients $ready\_ids$ at time $T$ from $Stat$
  **if** $ready\_ids$ is empty **then**
    Wait for another round to sample
  **else if** Number of clients in $ready\_ids < k$ **then**
    Sampled clients $round\_ids \leftarrow ready\_ids$
  **else**
    $round\_ids \leftarrow$ Sample $k$ clients from $ready\_ids$
  **for** client $ids$ in $round\_ids$ **do**
    (Interplay)
    Extract $ids$ device capacity $Dev_i$ from $Dev$
    Use $Dev_i$ to get its cost $t^i_{cost}$
    Update the $t_{unavail}$ part of Eq. (3) based on $Stat_i$
    **if** $ddl > t^i_{cost}$ **then**
      $\mathcal{S}^{ids} \leftarrow \mathcal{S}^{ids} + 1$
      $comm\_cost_i \leftarrow t^i_{cost}$
    **else**
      $comm\_cost_i \leftarrow ddl$
  $T \leftarrow T + \max_{i \in round\_ids} comm\_cost_i + t_{break}$
**return** $\mathcal{S}$ of each client

## Proofs

### Proof of Theorem 3.1

*Proof.* First, we note that Gaussian Mixture Models (GMM) can approximate any continuous distribution, given a sufficient number of kernels ([47], [42]). Therefore, GMM can model the distribution of speeds in the device database.

For any speed sample point $X$ in the database, the variance is given by:

$$\mathrm{E}\left[(X-\mu_0)^2 \mid \pi_1, \ldots, \pi_{K_n}, \mu_1, \ldots, \mu_{K_n}\right] \tag{4}$$

$$= \mathrm{E}\left[X^2 \mid \pi_1, \ldots, \pi_{K_n}, \mu_1, \ldots, \mu_{K_n}\right] - \mu_0^2 \tag{5}$$

$$= \sum_{k=1}^{K_n} \pi_k \left(\mathrm{E}\left[X_k^2 \mid \pi_1, \ldots, \pi_{K_n}, \mu_1, \ldots, \mu_{K_n}\right]\right) - \mu_0^2 \tag{6}$$

$$= \sum_{k=1}^{K_n} \pi_k \left(\sigma_k^2 + \mu_k^2\right) - \mu_0^2 \tag{7}$$

Where

$$\pi_1, \cdots, \pi_{K_n} \sim \text{Dir}\left(\frac{\alpha}{K_n}, \cdots, \frac{\alpha}{K_n}\right)$$

so

$$E(\pi_k) = \frac{\alpha}{K_n} \tag{8}$$

and

$$\mu_k \sim \mathcal{N}\left(\mu_0, (\sigma \cdot \sigma_0)^2\right)$$

so

$$E(\mu_k^2) = \mu_0^2 + (\sigma \cdot \sigma_0)^2 \tag{9}$$

Then, we have:

$$E(X - \mu_0)^2 = E\left[E\left[(X-\mu)^2 \mid \pi_1, \ldots, \pi_{K_n}, \mu_1, \ldots, \mu_{K_n}\right]\right] \tag{10}$$

$$= E\left[\sum_{i=1}^{K_n} \pi_i(\sigma_k^2 + \mu_k^2) - \mu_0^2\right] \tag{11}$$

$$= \sum_{k=1}^{K_n} E\left[\pi_k(\sigma_k^2 + \mu_k^2) - \mu_0^2\right] \tag{12}$$

$$= \sum_{k=1}^{K_n} \left[E(\pi_i)(\sigma_k^2 + E(\mu_k^2)) - \mu_0^2\right] \tag{13}$$

$$= \sum_{k=1}^{K_n} \left[\frac{\alpha}{K_n}(\sigma_k^2 + \mu_0^2 + (\sigma \cdot \sigma_0)^2) - \mu_0^2\right] \tag{14}$$

$$= \frac{\alpha}{K_n} \sum_{k=1}^{K_n} \sigma_k^2 + (\alpha - K_n)\,\mu_0^2 + K_n\,(\sigma \cdot \sigma_0)^2 \tag{15}$$

Considering this expression as a function of $K_n$ and $\sigma$, the range of the variance is given by:

$$(\min_{K_n}\left(\tfrac{\alpha}{K_n}\textstyle\sum_{k=1}^{K_n}\sigma_k^2 + (\alpha - K_n)\,\mu_0^2\right), +\infty).$$

Thus, for different specified sets of $K_n$ and $\sigma$, the variance of the sampled database can cover this range, which completes the proof. □

### Proof of Theorem 3.2

*Proof.* Let $X_i$ denotes the single state metric Metric (see Appendix Sec. V) of $D_{(i)}$, with $X_1 > \cdots > X_N$. For any state trace sample point $X$ in the database, the mean and variance of its single state metric $X$ are given by:

$$E(X \mid \alpha) = \sum_{i=startRank}^{N} \pi_i X_i \tag{16}$$

$$Var(X \mid \alpha) = \sum_{i=startRank}^{N} \pi_i (X_i - \sum_{i=startRank}^{N} \pi_i X_i)^2 \tag{17}$$

where

$$\pi_i = b_i \cdot (1 - \sum_{j=startRank}^{i-1} \pi_j)$$

Here, $b_i$ is a sample from the Beta distribution $beta(1, \alpha)$, implying that $E(b_i) = \frac{1}{1+\alpha}$. A smaller $\alpha$ leads to a concentration of the $\pi_i$ with samller $i$ values.

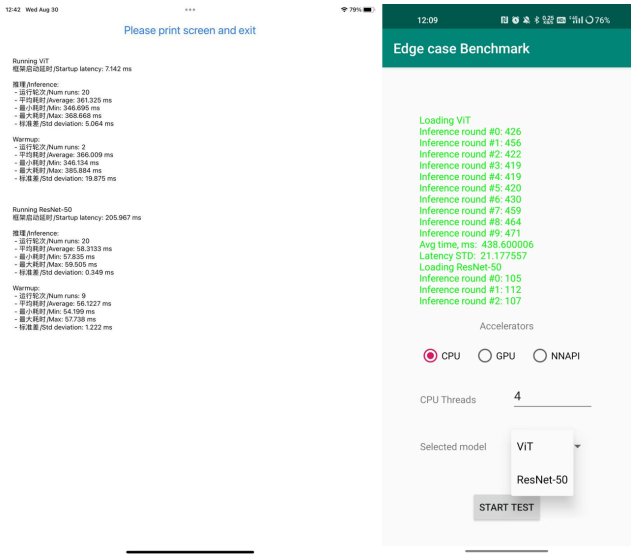



Figure S2. Overview of our APP.

When the range of $\alpha$ is (0, $+\infty$), and the range of $startRank$ is (1,$N$), the mean of the single state metric of the state database ranges from $X_N$ to $X_1$. Meanwhile, the variance of the single state metric of the state database ranges from 0 to $(X_1 - X_N)^2$. □

## II. Training Latency Data Collection APP for Mobile Devices

While current AI-Benchmark [22] provides latency information for most popular CNN architectures on Android devices, it lacks data for iOS devices and the recent Vision Transformers architectures. We propose two apps for iOS and Android devices based on TensorFlow Lite [45], fill this gap by capturing the latency computational information for iOS devices and recent Vision Transformer architectures, see Fig. S2 for an overview of our proposed APPs.

Our data collection procedure first run several inference rounds as warm-up before formal benchmarks, aiming to avoid the effects of loading native libraries and the boot latency of devices. After the warm-up, for each model, we randomly generate an input image with the shape of $224 \times 224$, run the inference process for 100 iterations, and record the total training latency. The apps then displays all latency results for collection.

Please note that although FLASH [60] and FedScale [29] have also provided similar real-world device databases, FLASH's database is relatively small containing only 19 devices. Additionally, the network uploading/downloading speeds in FedScale were measured as a single-point data format, which is insufficient for capturing the potential network fluctuations during FL communication rounds.

## III. Experimental Details

All the methods are implemented with Pytorch and optimized with SGD. All experiments are conducted on either a TITAN XP GPU or GeForce RTX 3090. Models used in this paper are all pretrained on ImageNets-1k [14]. We use 3 different random seeds for each experiment to mitigate occasionality.

In FL training, we set local training epoch to 1. Each round,

we randomly sample 20 ($k = 20$) clients per round. The total communication rounds are set to 3,000 for OpenImage and 4,500 for COVID-FL in deadline-based strategy, unless otherwise stated. We set the target accuracy of COVID-FL/OpenImage to 80%/25% in validation of device metrics (Sec. 5.2.1) and 75%/20% in validation of both state metrics and interplay metrics (Secs. 5.2.2 and 5.2.3) for readiness-based strategy.

We require a minimum selection of 2 clients per round. If this is not met, the round is marked as a failure, and the next round begins. We also ensure a minimum success ratio of 0.1, meaning that each FL training round is considered successful and the model is updated only if at least $0.1 \times k$ clients successfully complete the round.

In experiments to validate state metrics with real FL, we assume all clients use the same device with its $|\theta|/B_{up} + |\theta|/B_{down}$ as $\sim \mathcal{N}(\frac{ddl}{1.05}, \frac{ddl}{1.05} \times 0.1)$. This helps mitigate the impact of device heterogeneity. We use $ddl = 120$ for deadline-based strategy, and uniformly use seconds as the unit for all time-related data. We set specified proportion rate $= 1.0$ for readiness-based strategy in our experiments, unless otherwise stated.

**Hyperparameters for compared FL methods.** In FedProx [31], $\mu$ is set to 0.1 and 0.01 for OpenImage and COVID-FL datasets, respectively. In FedDyn [5], the $\alpha$ is set to 0.01. In FedKD [55], the energy thresholds $T_{start}$ and $T_{end}$ are 0.95 and 0.98. In FedBuff [36], we set the server momentum $\beta$ to 0.9 and $K$ to 10. In FetchSGD [40], $k$ for sketch is set to 100000 and $\beta$ for global momentum is set to 0.9.

**Generation of heterogeneous device databases for validation of our device metrics in Sec. 5.2.1.** Our DPGMM and DPCSM allows genearting arbitrary heterogeneity by manipulating parameters. For generating a wide range of heterogeneous device database, we apply our DPGMM approach and set $K_n$ to $\{50, 100, 200\}$ and $\sigma$ to $\{0.1, 0.4, 0.7, 1.1, 1.4, 1.7, 2.0\}$, with an average device speed of 6 MB/s, to sample 21 sets of device databases from our base device dataset. Each set contains 100 devices. We illustrate the sampling results of device speeds in Fig. S3.

**Generation of heterogeneous state databases for validation of our state metrics in Sec. 5.2.2.** For heterogeneous state database, we implement DPCSM with $\alpha$ in $\{1, 2, 3, 4, 5, 10, 100\}$ and $StartRank$ in $\{0, 100, 350\}$. Each combination of $\alpha$ and $StartRank$ generates state databases consisting of 100 states. Wide range of $\alpha$ and $StartRank$ promise the different levels of ST-Client-Ratio and diversity of sampled state databases.

**Generation of heterogeneous device and state databases for validation of our interplay metrics in Sec. 5.2.3.** We independently extract four sets of device and state databases from above sampled results, ranging from mild to severe heterogeneity. Their corresponding metrics are DevMC-R $\in \{0.806, 0.896, 0.918, 0.928\}$ and StatMC-R $\in \{0.752, 0.778, 0.858, 0.956\}$. Each set consists of 100 devices and 100 states. By creating all potential pairwise combinations, we generate 16 distinct interplay device and state sets, representing diverse real-world FL scenarios with varying device and state heterogeneity.

In Sec. 5.3 we use the same 16 distinct interplay device and state sets for evaluating current FL algorithms.

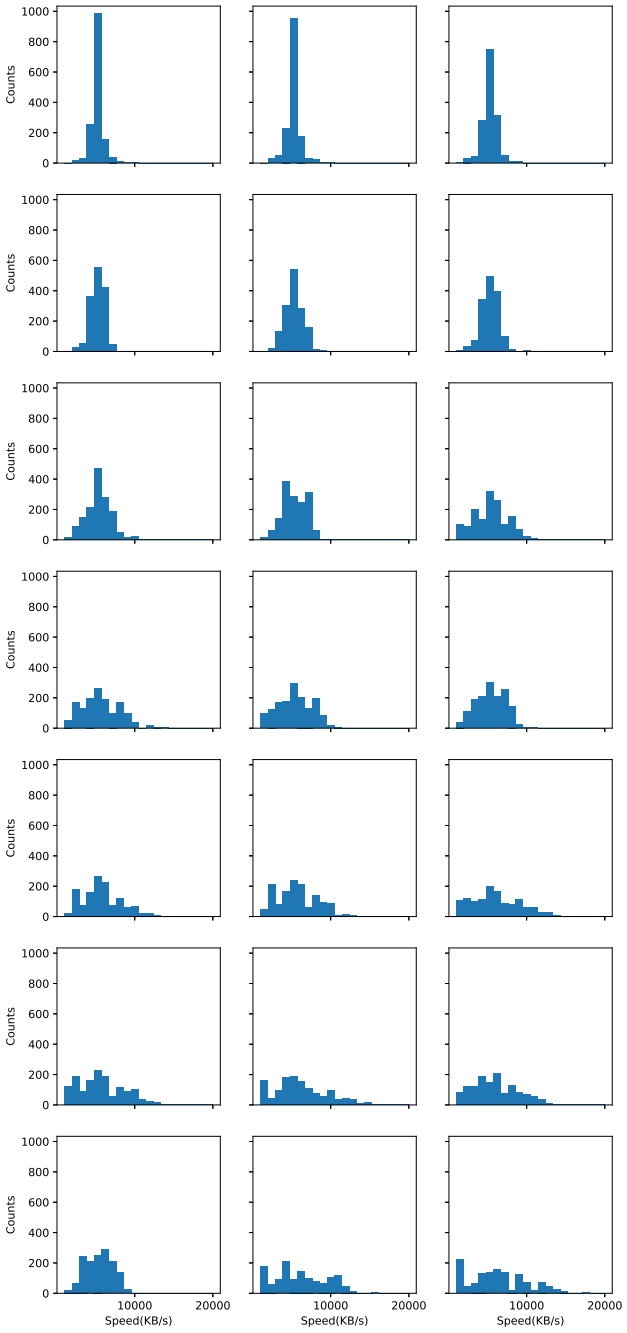
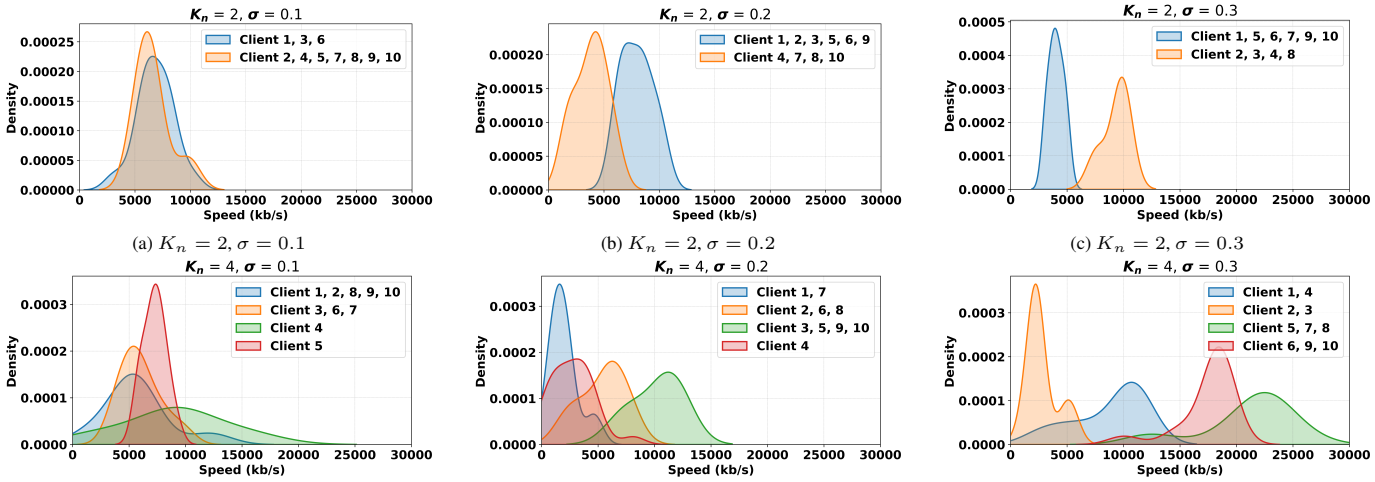



Figure S3. Distribution of device speeds used in device heterogeneity experiments. For simplicity, we only show the network downloading speeds here.

## IV. Ablation Study of BiasPrompt+

To demonstrate the effectiveness of gradient surgery-based staleness-aware aggregation strategy and communication-efficient module BiasPrompt via fast weights in BiasPrompt+, we conduct an ablation study. BiasPrompt represents the method that that solely incorporates our BiasPrompt module. The staleness-aware strategy is represented by FedAVG+staleness-aware, which indicates adding the staleness-aware strategy to FedAVG. BiasPrompt+ signifies the adoption of both the gradi-




(a) $K_n = 2, \sigma = 0.1$ (b) $K_n = 2, \sigma = 0.2$ (c) $K_n = 2, \sigma = 0.3$

(d) $K_n = 4, \sigma = 0.1$ (e) $K_n = 4, \sigma = 0.2$ (f) $K_n = 4, \sigma = 0.3$

Figure S4. Results of DPGMM Sampling under different parameter settings.

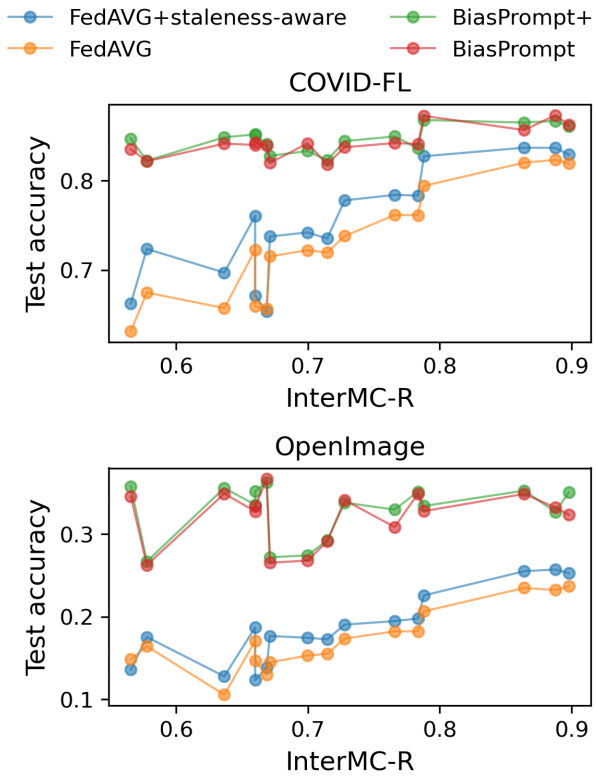



Figure S5. Ablation study for BiasPrompt+ on OpenImage and COVID-FL. FedAVG+staleness-aware indicates the FedAVG with the adoption of the gradient surgery-based staleness-aware aggregation strategy. BiasPrompt+ implements both gradient surgery-based staleness-aware aggregation strategy and communication efficient module BiasPrompt.

ent surgery-based staleness-aware aggregation strategy and the communication-efficient module BiasPrompt via fast weights. As the staleness-aware aggregation strategy is specifically designed for deadline-based approaches, we only report the test accuracy results for COVID-FL and OpenImage datasets within deadline-based strategies. All other experimental settings remain consistent with those described in Sec. 5.3.

Fig. S5 illustrates the test accuracy of various combinations of our proposed modules. From the figure, we can observe that BiasPrompt module greatly improves the performance compared to FedAVG. This is because BiasPrompt tackles the long wall-clock issue by its communication efficiency. In addition, FedAVG+staleness-aware outperforms FedAVG in most heterogeneity cases. The staleness-aware strategy in FedAVG+staleness-aware allows stale clients to participate in the aggregation, thereby greatly improving the utilization of their updates. The gradient surgery-based approach ensures that stale updates do not negatively impact the fresh updates, allowing the server model to receive more generalized updates while maintaining the correct gradient direction of the current model state.

BiasPrompt+ achieves the best accuracy among the ablation methods, signifying that combining the communication-efficient module BiasPrompt and gradient surgery-based aggregation strategy effectively reduces wall-clock time, increases client participation, and mitigates risks from stale gradients.

## V. Single State Metric for DPCSM

The single state metric, which is also called ST-Client-Ratio, can be modified from StatMC-R. Specifically, given the baseline dataset of $N$ states and the specified $ddl$, we conduct a Monte Carlo simulation of a real FL training process with these states and use Algorithm S4 to derive the number of successful participation times $\mathcal{S}_i$ for each state. Using this information, we calculate the single metric for the $i_{th}$ state $(i = 1, \cdots, N)$ by comparing the simulated number of successful participation times of each state (which accounts for state heterogeneity) with the ideal number of successful participation times $\mathcal{S}_{ideal}$ (which assumes a homogeneous setting) as:

$$\int_0^{ddl} \frac{Clip(\mathcal{S}_i, 0, \mathcal{S}_{ideal})}{\mathcal{S}_{ideal}} p\left(t^i_{cost}\right) dt^i_{cost}, \tag{18}$$

where $t^i_{cos}$, $S_{ideal}$ are defined the same as in StatMC-R.

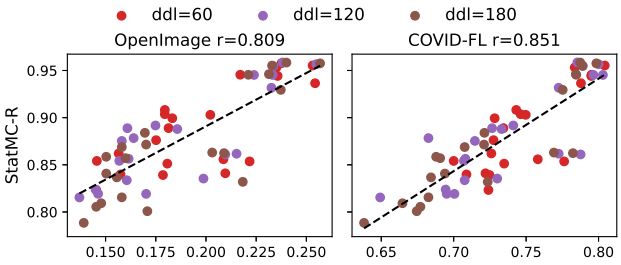



Figure S6. StatMC-R vs. test accuracy with 3 different *ddl* on COVID-FL/OpenImage using deadline-based strategy.

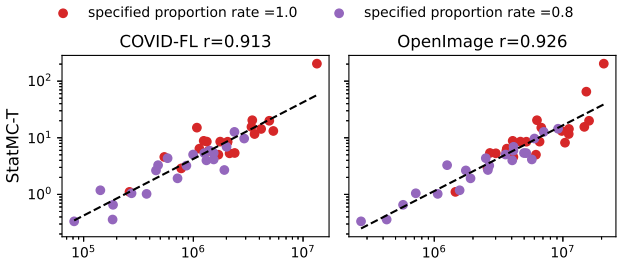



Figure S7. StatMC-T vs. training time with 2 different specified proportion rates on COVID-FL/OpenImage readiness-based strategy.

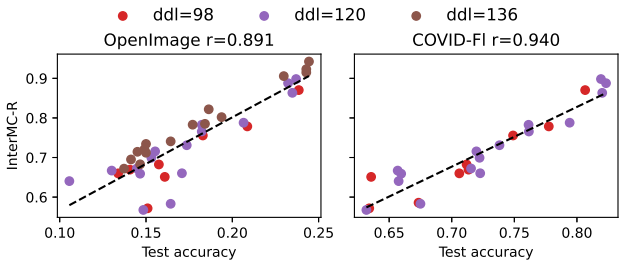



Figure S8. InterMC-R vs. test accuracy with 3 different *ddl* on COVID-FL/OpenImage using deadline-based strategy.

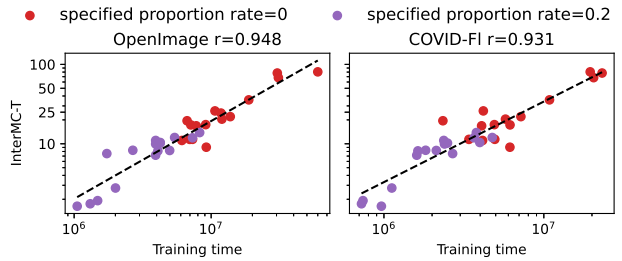



Figure S9. InterMC-R vs. training time with 2 different specified proportion rates on COVID-FL/OpenImage readiness-based strategy.

# VI. Additional Results

## VI.1. Additional Results of State and Interplay Heterogeneity Metrics

Similar to device heterogeneity metrics, *ddl* and specified proportion rate are two important factors that affect both FL performance and metric values. We further explore the alignment of our state and interplay metrics across different *ddl* and specified proportion rate.

**Additional validation experiments for state metrics:** For state metrics, we conduct additional experiments with $ddl \in \{60, 120, 180\}$ and specified proportion rate $pr \in \{1, 0.8\}$. Here we use different *ddl* sets with larger intervals than those used in device heterogeneity experiments to make results more discriminatory. As shown in Fig. S6, even with 3 different *ddl* settings, StatMC-R consistently exhibits a high correlation with prediction accuracy of FedAVG on COVID-FL and OpenImage. Moreover, it is noteworthy that StatMC-T exhibit a higher Pearson correlation coefficient $r$ ($> 0.91$) when using 2 different specified proportion rates, as depicted in Fig. S7. These results further demonstrate our metrics' effectiveness and scalability.

**Additional validation experiments for interplay metrics:** For interplay metrics, we conduct additional experiments with $ddl \in \{98, 120, 136\}$ and specified proportion rate $pr \in \{1, 0.8\}$. Here *ddl* uses the same *ddl* sets as in device heterogeneity experiments, as it can cover a wide range of cases. As depicted in Fig. S8, as *ddl* increases, the test accuracy improves with InterMC-R, exhibiting strong consistency with high Pearson correlation coefficient $r$ ($> 0.89$). Similarly, when the specified proportion rate is set to 0.8 in Fig. S9, the metrics still exhibit strong correlation ($r > 0.93$), indicating their robustness in assessing the impact of heterogeneity on FL performance.

In summary, these additional validation experiments demonstrate the scalability and effectiveness of our state and interplay metrics across different *ddl* values and specified proportion rates, capturing key attributes affecting real-world FL performance. With our metrics, we believe that FL designers can comprehensively assess device/state heterogeneity and address related issues, ultimately leading to the development of more robust and efficient FL systems.

# VII. Generalization to Other Baseline Network and Datasets

Effective device and state heterogeneity assessing metrics should be independent of baseline network and dataset. Here, we conduct experiments to validate the generalization of our metrics on an extra Cifar10 [28] dataset using ViT, and on OpenImage, COVID-FL, and Cifar10 dataset using ResNet50 [19] as baseline network.

## VII.1. Generalization to Cifar10

For the Cifar10 dataset, we randomly assign 10 images to each client's local training set, and use the original test set as our global test set. We set the total communication rounds to 3000 in deadline-based strategy and set the target accuracy to 97% in validation of device metrics and 95% in validation of both state metrics and interplay metrics for readiness-based strategy. Other experimental settings remain consistent with those in Sec. 5.2. We present the results in Fig. S10 and Fig. S11. We observe strong correlations ($r > 0.81$) which indicate our metrics' scalability to other datasets.

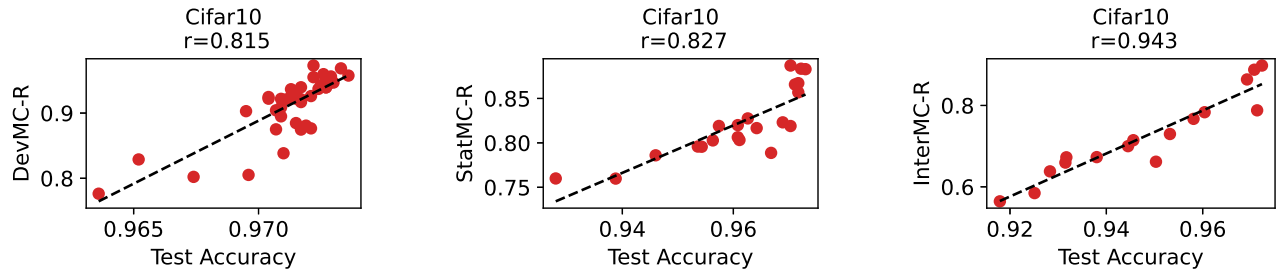



Figure S10. DevMC-R, StatMC-R and InterMC-R vs. test accuracy on Cifar10 with ViT.

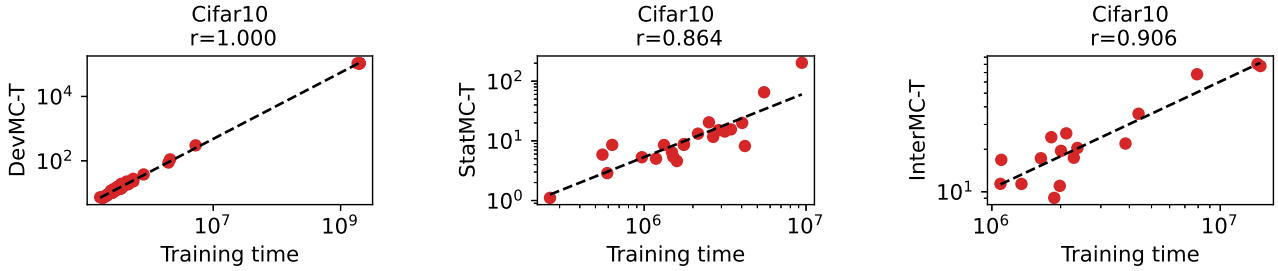



Figure S11. DevMC-T, StatMC-T and InterMC-T vs. training time on Cifar10 with ViT.

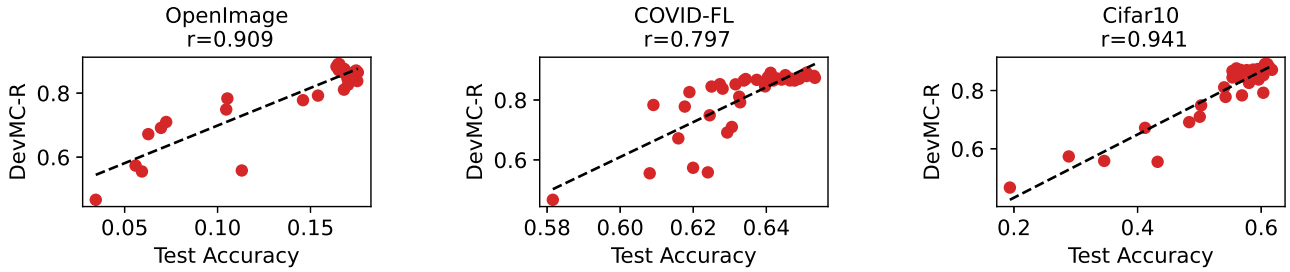



Figure S12. DevMC-R vs. test accuracy on OpenImage, COVID-FL, and Cifar10 with ResNet50.

## VII.2. Generalization to ResNet50

To verify the scalability of our metrics across network architectures, we then change the baseline network from ViT used in the paper to ResNet50 and conduct experiments across OpenImage, COVID-FL and Cifar10 datasets. We scale the *ddl* to $\{26, 28, 36\}$ according to the model size and keep other experimental settings the same as in Sec. 5.2. The results are presented in Fig. S12-S17. Remarkably, our device, state, and interplay metrics continue to exhibit a strong correlation with FL performance under different network structures and datasets. The Pearson correlation coefficient $r$ consistently surpass 0.77 for both deadline-based strategy and readiness-based strategy, indicating the irrelevance of our metrics to the network structure.

## VII.3. Comparison to Other FL Methods

We compare BiasPrompt+ to the latest SOTA TimelyFL [66] on OpenImage. We use the same device and state datasets used in Tab. 2. For TimelyFL, we assume that the device speed is known and update the partial layers based on *ddl* and the current device speed. As depicted in Tab. S1, BiasPrompt+ outperforms TimelyFL for all degrees of heterogeneity and strategies. In addition, TimelyFL also struggles with increased device and state heterogeneity, which may be due to the fact that only a small fraction of the network parameters are updated in order to maintain client participation rates.

Table S1. Test accuracy↑ and training time↓ (separated by /) on OpenImage. We compare BiasPrompt+ with the latest TimelyFL using ViT. We also show the results of BiasPrompt+ using ResNet50 (R in Table) and compare it to FetchSGD.

| InterMC-R | BiasPrompt+ | TimelyFL |
|---|---|---|
| 0.56 (severe) | **35.75/8.6h** | 12.86/97h |
| 0.73 (moderate) | **33.79/2.2h** | 18.98/78h |
| 0.86 (mild) | **35.24/1.4h** | 24.41/61h |

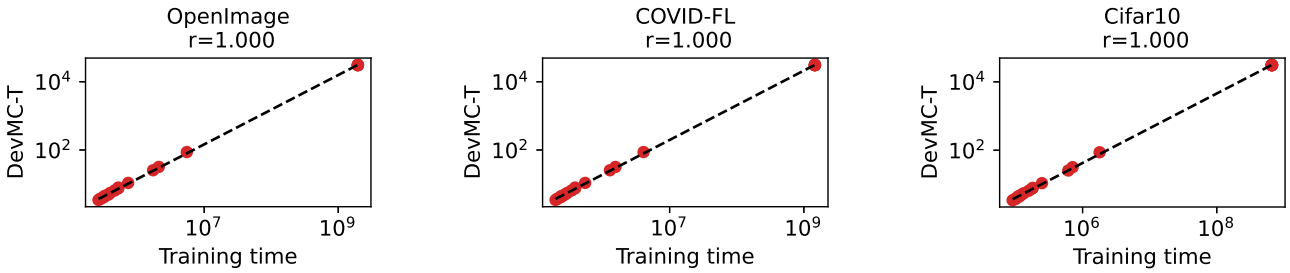



Figure S13. DevMC-T vs. training time on OpenImage, COVID-FL, and Cifar10 with ResNet50.

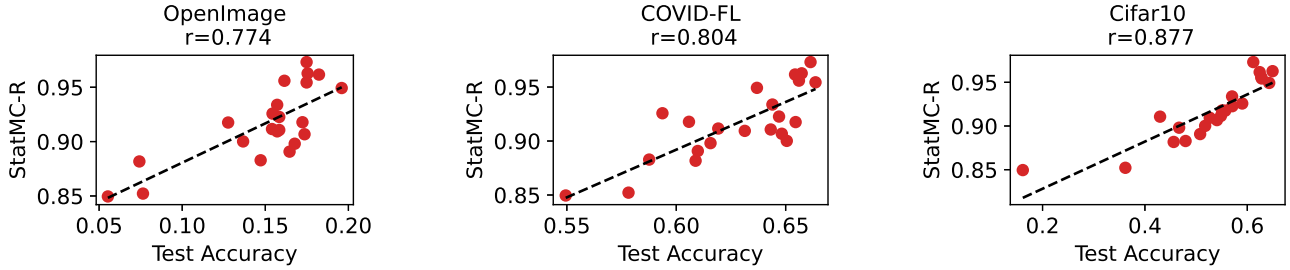



Figure S14. StatMC-R vs. test accuracy on OpenImage, COVID-FL, and Cifar10 with ResNet50.

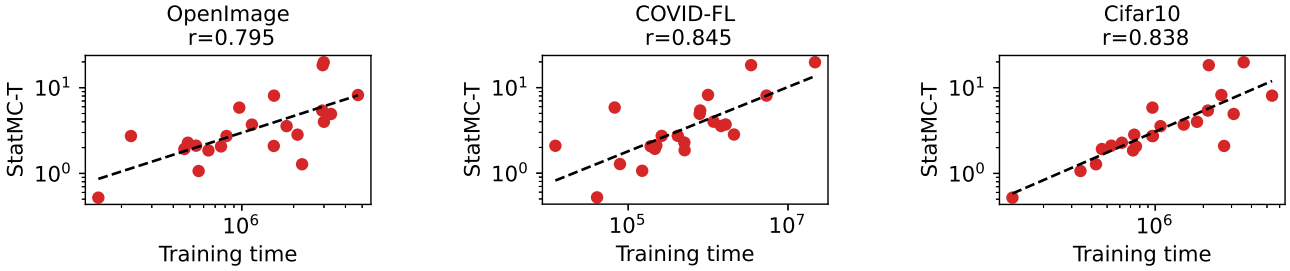



Figure S15. StatMC-T vs. training time on OpenImage, COVID-FL, and Cifar10 with ResNet50.

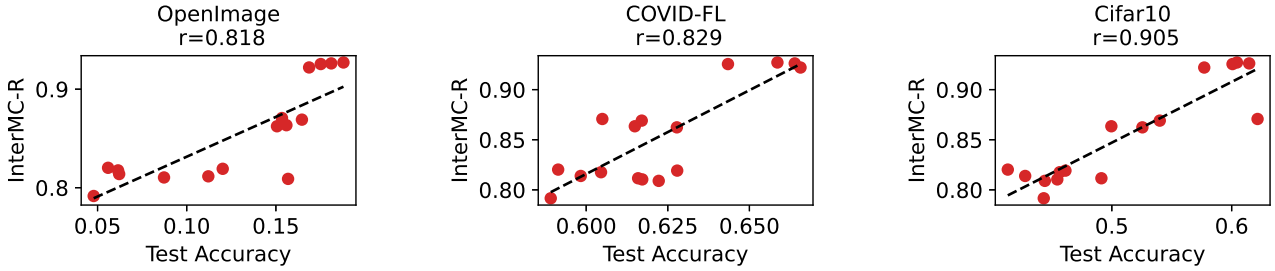



Figure S16. InterMC-R vs. test accuracy on OpenImage, COVID-FL, and Cifar10 with ResNet50.

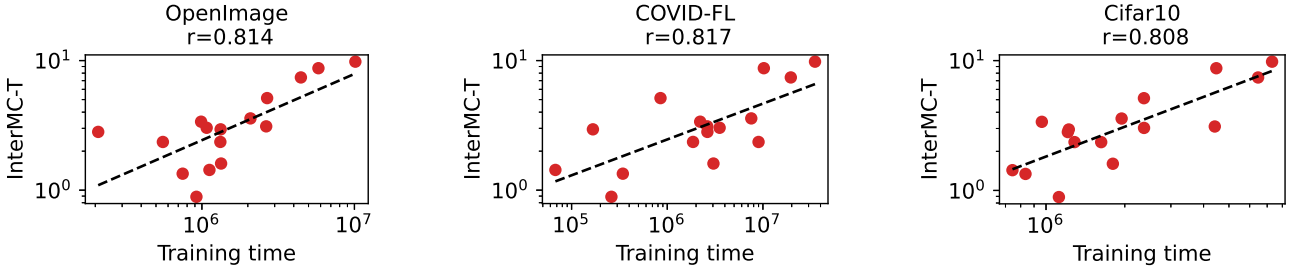



Figure S17. InterMC-T vs. training time on OpenImage, COVID-FL, and Cifar10 with ResNet50.