EcoLearn: Optimizing the Carbon Footprint of Federated Learning

Talha Mehboob
University of Massachusetts
Amherst, USA

Noman Bashir
Massachusetts Institute of
Technology, USA

Jesus Omana Iglesias
Telefonica Research, Spain

Michael Zink
University of Massachusetts
Amherst, USA

David Irwin
University of Massachusetts
Amherst, USA

Abstract

Federated Learning (FL) distributes machine learning (ML) training across edge devices to reduce data transfer overhead and protect data privacy. Since FL model training may span hundreds of devices and is thus resource- and energy-intensive, it has a significant carbon footprint. Importantly, since energy's carbon-intensity differs substantially (by up to 60×) across locations, training on the same device using the same amount of energy, but at different locations, can incur widely different carbon emissions. While prior work has focused on improving FL's resource- and energy-efficiency by optimizing time-to-accuracy, it implicitly assumes all energy has the same carbon intensity and thus does not optimize carbon efficiency, i.e., work done per unit of carbon emitted.

To address the problem, we design `EcoLearn`, which minimizes FL's carbon footprint without significantly affecting model accuracy or training time. `EcoLearn` achieves a favorable tradeoff by integrating carbon awareness into multiple aspects of FL training, including i) selecting clients with high data utility and low carbon, ii) provisioning more clients during the initial training rounds, and iii) mitigating stragglers by dynamically adjusting client over-provisioning based on carbon. We implement `EcoLearn` and its carbon-aware FL training policies in the Flower framework and show that it reduces the carbon footprint of training (by up to 10.8×) while maintaining model accuracy and training time (within ~1%) compared to state-of-the-art approaches.

SEC '25, Arlington, VA, USA

ACM ISBN 979-8-4007-2238-7/2025/12
https://doi.org/10.1145/3769102.3770625

CCS Concepts

• **Computer systems organization** → **Distributed architectures**; **Client-server architectures**.

Keywords

Sustainability; carbon footprint; federated learning

ACM Reference Format:
Talha Mehboob, Noman Bashir, Jesus Omana Iglesias, Michael Zink, and David Irwin. 2025. EcoLearn: Optimizing the Carbon Footprint of Federated Learning. In *The Tenth ACM/IEEE Symposium on Edge Computing (SEC '25), December 3–6, 2025, Arlington, VA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3769102.3770625

1 Introduction

Federated Learning (FL) is an increasingly popular machine learning (ML) approach that trains models on data distributed across numerous edge devices [1, 2], potentially spread over large geographic areas. FL enhances data privacy and security compared to centralized ML by processing data locally and only sending updated model parameters to a central server. This reduces data transfer overhead and keeps raw data on the device, safeguarding user privacy. In many application scenarios, FL is an important tool for complying with data protection laws, such as GDPR [3].

Unlike centralized learning, FL operates on distributed data across many clients. In each training round, only a subset of clients is selected, often randomly. These clients perform multiple training iterations, e.g., using stochastic gradient descent on their local data, before sending model updates to a central server for aggregation into the global model. FL models are typically trained on privacy-sensitive, heterogeneous data from diverse environments. For example, images from different CCTV cameras in a hospital parking lot on opposite coasts in the United States during winter may vary significantly, meaning data across clients may not be independent and identically distributed (i.i.d.). Importantly,

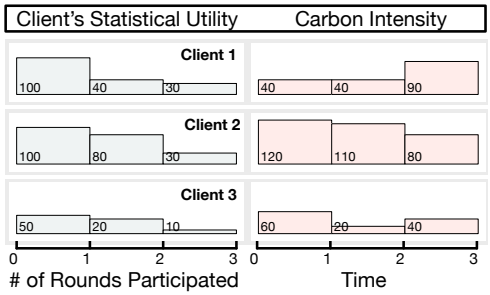


Figure 1: ***Client's statistical utility decreases as the client participates in training (shown in left), whereas the carbon intensity for each client varies over time (shown in right). One unit of time is one training round.***

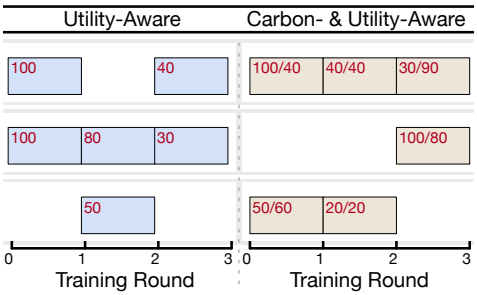


Figure 2: ***The utility-aware baseline greedily selects clients with the highest utility resulting in carbon cost of 460 (left). The utility- and carbon-aware policy selects using carbon-normalized utility and has a carbon cost of 330 (right).***

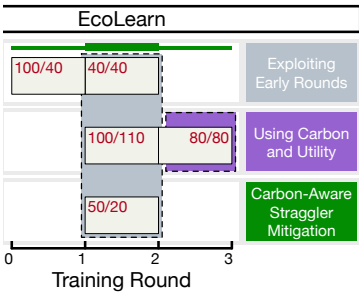


Figure 3: **EcoLearn** ***selects high-utility, low-carbon clients early to exploit the critical learning period and does carbon-aware straggler mitigation, yielding carbon cost of 290.***

the clients selected in each round can impact i) the time required to reach a target accuracy or ii) the accuracy achieved within a fixed training time. For example, repeatedly selecting clients missing certain data classes may prolong training, requiring additional rounds to cover those classes.

FL often selects random subsets of clients each round. However, this approach can increase training time by requiring many rounds to ensure sufficient data coverage. Random selection may also waste resources by continually training on data that contributes little to model accuracy. Recent research focuses on intelligent participant selection, which chooses clients based on their data's statistical "utility" – how much data improves the global model's accuracy [4, 5]. Such intelligent client selection significantly reduces resource waste and shortens training time. Unfortunately, while reducing the training time necessary to achieve a target accuracy is important, prior work generally does not consider that clients may incur widely different "costs" for local training. In particular, an increasingly important cost is energy's carbon intensity, which may differ widely across geographically distributed clients due to varying sources of electric energy (e.g., fossil fuels, renewables, nuclear, etc.) in different locations. Thus, optimizing a model's carbon footprint is becoming just as important as optimizing its accuracy and training time.

Recent work [6] suggests that by 2026 [7], advancements in foundational models will increasingly rely on FL, as organizations with computational resources but restrictive data policies opt for collaborative training [8–10]. This shift has significant sustainability implications, as training a small ML task using FL can generate higher carbon emissions than training large transformer models in a centralized setting [11]. These challenges emphasize the need for research to enhance efficiency and sustainability in FL.

To address the problem, we present EcoLearn, which concurrently optimizes FL's carbon footprint, accuracy, and training time. As we discuss, gracefully navigating this three-way tradeoff requires integrating carbon awareness into multiple aspects of FL training, such as deciding which clients to select, how many to select, and when. As shown in Figure 1 and Figure 2, existing utility-based (and carbon-agnostic) client selection achieves a high model accuracy and low training time but incurs a high carbon cost [4, 5]. Alternatively, a simple carbon-only policy that selects only low-carbon clients significantly increases training time and decreases accuracy due to low data coverage (not shown in this figure).

EcoLearn optimizes its carbon footprint while maintaining training time and accuracy by integrating carbon-awareness into *when*, *how many*, and *which* clients to select. EcoLearn exploits the "neuroplasticity" of a model in early rounds, i.e., its *critical learning period* [12, 13], to bootstrap model weights using a large number of high- clients. EcoLearn improves accuracy beyond a simple utility-based approach by selecting only high-utility clients during this early-round scale-up. However, it selects clients with high normalized utility per unit carbon, sacrificing the potential increase in accuracy beyond the utility-based approach to lower its carbon footprint, as shown in Figure 3. The final ingredient is dynamic over-provisioning of clients for straggler mitigation, selecting more "extra" clients during low carbon periods to reduce training time, instead of a fixed ratio used by prior work [4]. Notably, making existing carbon-agnostic utility-based techniques carbon-aware is insufficient, as it reduces the accuracy to reduce the carbon footprint (shown in Section 4.2). This motivates EcoLearn's additional optimizations including time-varying scaling within critical learning period and incorporating carbon-awareness into straggler provisioning. EcoLearn's techniques can generalize to "cost" metrics other than carbon.

Our hypothesis is that EcoLearn's adaptive carbon-aware policies for which, when, and how many clients to select can substantially lower FL's carbon footprint (by an order of

magnitude) while maintaining model accuracy and training time, even for highly non-i.i.d. data, compared to state-of-the-art carbon-agnostic approaches.

In evaluating our hypothesis, we make the following contributions.

1 – Carbon Awareness in FL. Optimizing FL's carbon footprint by leveraging spatiotemporal variations in the grid's carbon intensity presents a fundamental tradeoff between carbon, model accuracy, and training time. A singular focus on selecting *which* clients for training cannot gracefully navigate this tradeoff. To do so, we integrate carbon awareness into multiple aspects of client selection (see Section 2).

2 – Practical Adaptive Carbon-Aware Policies. We design adaptive carbon-aware policies for `EcoLearn` that solve the challenges of precisely determining which, how many, and when to select clients when training an FL model in practice. `EcoLearn`'s policies lie at the Pareto-frontier of model accuracy and carbon cost to train the model (Section 3).

3 – Implementation and Evaluation. We implement `EcoLearn` using the Flower framework [14], and evaluate it across multiple synthetic and real datasets with different data distributions across clients. Our results show that `EcoLearn` reduces the carbon footprint of FL training (by up to 10.8×) while maintaining model accuracy and training time of state-of-the-art approaches (details in Section 4).

2 Background and Motivation

Below, we provide an overview of FL (Section 2.1), resource- and carbon-efficient FL (Section 2.2) and `EcoLearn`'s approach to carbon-efficient FL (Section 2.3).

2.1 Federated Learning Overview

Federated Learning (FL) is a machine learning (ML) approach for iteratively training a model where the training data is distributed across many client devices [2]. An FL framework comprises two main components: a centralized server or *aggregator* and distributed devices or *clients*. The training process occurs over multiple *rounds* to improve model accuracy by leveraging numerous clients [15].

In each round t, the aggregator determines *which* and *how many* clients will participate. Let $\mathcal{K}_t$ be set of clients in round t, and N be the total number of clients. Typically, $|\mathcal{K}_t| \ll N$ since only a small fraction of clients participate in each round due to communication bottlenecks. Each participating client $k \in \mathcal{K}_t$ receives the global model parameters $\mathbf{w}_t$ and trains on its local dataset D_k. The objective at each client is to minimize the local loss function $\mathcal{L}_k(\mathbf{w})$ on the local data D_k:

$$\mathcal{L}_k(\mathbf{w}) = \frac{1}{|D_k|} \sum_{i \in D_k} \ell(\mathbf{w}; x_i, y_i)$$

Table 1: List of variables and their descriptions.

Variable	Description
$\mathcal{K}_t$	Set of clients selected to participate in round t.
N	Total number of clients in the FL system.
$\mathbf{w}_t$	Global model parameters at round t.
D_k	Local dataset of client k.
$\mathcal{L}_k(\mathbf{w})$	Local loss function for client k, based on dataset D_k.
$\ell(\mathbf{w}; x_i, y_i)$	Loss function for a single data point (x_i, y_i).
$\Delta\mathbf{w}_k$	Model update (gradient) from client k.
η	Learning rate for gradient descent.
$\lvert D_{\text{total}}\rvert$	Total data points across all participating clients in a round.

where $\ell(\mathbf{w}; x_i, y_i)$ represents the loss for a single data point (x_i, y_i), and $\mathbf{w}$ denotes the model parameters.

The clients then compute the gradient update $\Delta\mathbf{w}_k = \mathbf{w}_t - \eta\nabla\mathcal{L}_k(\mathbf{w}_t)$ using their local data. After local training, each client sends the model update back to the server, which aggregates these updates. A common aggregation strategy is *Federated Averaging (FedAvg)* [2], which updates the global model as a weighted average of the clients' updates:

$$\mathbf{w}_{t+1} = \frac{1}{|\mathcal{K}_t|} \sum_{k \in \mathcal{K}_t} \frac{|D_k|}{|D_{\text{total}}|} \Delta\mathbf{w}_k$$

where $|D_{\text{total}}|$ is the total number of data points for all participating clients in that round.

However, some client updates may not arrive due to network issues, computational resource availability, or energy constraints. Clients that do not respond or take too long are called *stragglers*, and can significantly delay training. To mitigate these stragglers, the aggregator may use techniques such as discarding updates from late-arriving clients.

FL typically includes criteria that determine when to stop training [2, 16]. We focus on two widely-used criteria: *model convergence*, where training stops when gains in model performance become small, i.e., $||\mathbf{w}_{t+1} - \mathbf{w}_t||_2 < \tau$, where τ is the threshold, and *performance threshold*, where training halts when the model's accuracy on a validation dataset reaches a predefined threshold T_{perf}.

2.2 Carbon-Aware and Resource-Efficient Federated Learning

In this section, we review recent work on improving FL's resource efficiency. We then present an analysis of spatiotemporal variations in energy's carbon intensity across clients. We finally discuss how prior work on resource-efficient FL does not directly apply to optimizing carbon.

1 – Resource efficiency in FL. FL's resource efficiency is dictated by the operation of its aggregator and clients. Prior work has explored optimizing resource requirements for aggregators, e.g., [17]. However, clients incur most of the computational load in FL using their computational and communication resources. Prior work has advanced optimizing

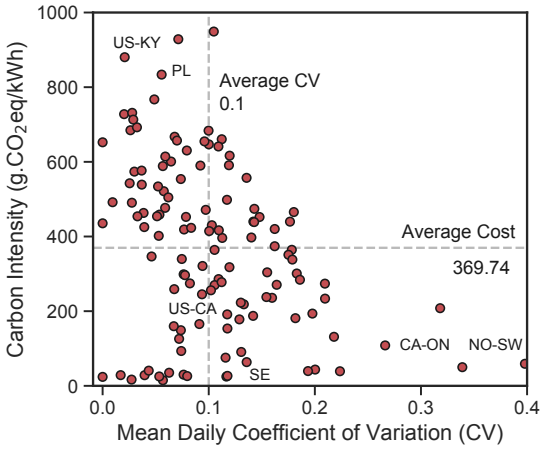


Figure 4: ***Average carbon intensity variations in different geographical regions expressed as the coefficient of variation (CV) of each region's carbon intensity. The dotted lines in the figure represent the average carbon intensity and the average CV across all regions.***

model performance [18] and system efficiency [2] to tackle the challenges posed by clients' data and system heterogeneity. However, further optimizations, such as *intelligent* client selection instead of the traditional random selection [15, 19], have only recently received attention.

Recent work has improved on random selection. For instance, Oort [4] employs a guided client selection approach to prioritize clients with higher statistical utility to maximize system efficiency. Statistical utility is measured using training loss as a proxy, while system efficiency is determined by completion time. Oort prioritizes faster clients to reduce the round training time. It uses a pacer algorithm, that allows for longer round durations to include slower or unexplored clients, thereby improving the model's overall statistical efficiency. REFL [5] extends Oort to minimize resource waste by incorporating model updates from straggler clients as they arrive after each round ends. Other approaches, such as PyramidFL [20], fine-tune client selection by considering data heterogeneity within selected clients and data plus system heterogeneity between selected and non-selected clients.

2 – Spatiotemporal variations in carbon intensity. FL's carbon footprint depends on energy's carbon intensity, measured in grams of CO_2 equivalent per kilowatt-hour ($g{\cdot}CO_2eq/kWh$), across geographically uniformly distributed clients. Due to differences in energy sources—ranging from fossil fuels like coal and gas to renewables like hydro and wind—energy's carbon intensity can vary significantly between client locations. Figure 4 shows a scatterplot of the average carbon intensity against its daily coefficient of variation, computed as the standard deviation over the mean. Each circle represents one of the 100 locations worldwide. As shown, there is a significant difference in carbon intensity across locations (along the y-axis) and high variations over a single day (along the x-axis) in a given location. These ***real-world spatiotemporal variations in carbon intensity demonstrate a significant potential for carbon-aware federated learning***.

We use real carbon-intensity data from ElectricityMap [21], a third-party service that provides aggregated real-time data on grid electricity carbon intensity, to model and derive client carbon costs in our experiments. Another widely used service is WattTime [22], which also offers real-time carbon intensity data. These sources provide the carbon intensity data for up to 99% of geographical locations globally, with only a few exceptions (e.g., parts of far-eastern Russia, Svalbard, and Jan Mayen). Companies such as Google already rely on these services to inform users of carbon intensity estimates for their cloud regions [23]. Thus, intelligent carbon-aware client selection selects clients only if their carbon intensity is low, reducing the carbon cost incurred by each client and, ultimately, FL's global carbon footprint.

3 – Resource-efficient FL for carbon optimizations. Prior work defines resource efficiency in terms of the time to reach a target model accuracy. This may not always translate to reducing carbon footprint, as participating clients may have high carbon during that time. Also, the work on reducing wasted computations at stragglers, at best, reduces system-level carbon waste but does not impact the carbon emitted by each client. Prior work such as Oort can be extended to reduce carbon by treating carbon as a resource or a cost and maximizing 1/carbon instead of utility. However, in Section 4, we show that extending existing resource-efficient FL approaches for carbon awareness by simply changing their objective to carbon does not result in a desirable tradeoff between carbon, training time, and model accuracy.

Optimizing FL's carbon footprint instead requires integrating carbon awareness not only in the decision of *which* clients to select but also *how many* to select and *when*. We now present the FL model training insights that `EcoLearn` leverages to inform its decisions.

2.3 Key Ingredients of `EcoLearn`

`EcoLearn` decomposes the problem of carbon-efficient client selection into three key steps: first, determining *when* and *how many* clients to select during the training duration; second, deciding *which* clients to select in each round of training; and third, *how many* replica tasks to prevision in order to mitigate straggler-induced delays.

In making these decisions carbon-aware, `EcoLearn` exploits fundamental insights from prior work on how the utility of clients in improving global model accuracy can change based on data heterogeneity, the number of rounds progressed, and the number of clients selected in each round.

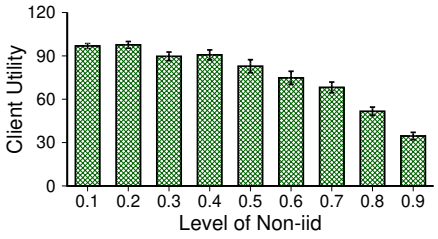


Figure 5: Impact of data heterogeneity across clients on the statistical utility of a given client toward the global federated learning model.

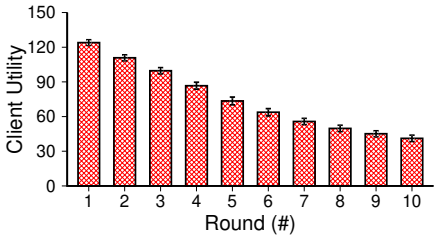


Figure 6: Impact of the number of training rounds a client participates in on the statistical utility of a client at a fixed level of non-iidness.

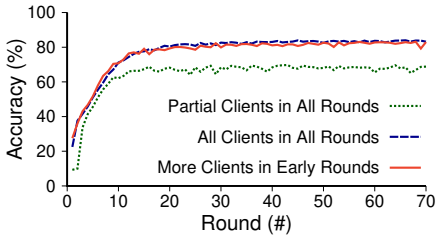


Figure 7: Impact of how many clients are selected across rounds on the accuracy of the globally trained federated learning model.

Table 2: Summary of related FL frameworks and their key attributes compared to EcoLearn.

FL Framework	Client Selection	Straggler Mitigation	Client Scaling	Carbon Awareness
Oort	✓	✓	✗	✗
REFL	✓	✓	✗	✗
CriticalFL	✗	✗	✓	✗
FedZero	✓	✓	✗	✓
PyramidFL	✓	✓	✗	✗
EcoLearn	✓	✓	✓	✓

1 – Variations in utility across clients. A client's statistical utility quantifies its importance in improving model accuracy [4]. Figure 5 shows a client's statistical utility as a function of data heterogeneity across clients (quantified using the level of non-iid). In general, if all clients have the same distribution of classes, the data is independent and identically distributed (iid). If clients have different distributions of classes, it is considered non-iid. Section 4 describes how we compute the level of non-iid for a given dataset.

As shown in Figure 5, the average utility of participating clients decreases as the level of non-iid increases; model updates from clients with highly specific data are less useful. Instead of blindly minimizing carbon cost, EcoLearn opportunistically trades carbon footprint for gain in model accuracy, i.e., it accepts a high carbon cost for a client with high utility. More formally, *it selects clients based on their normalized utility per unit of carbon.*

2 – Change in utility over rounds. If a model has learned from a given dataset, even partially, the value of learning on that data reduces. Figure 6 shows that as clients participate in training rounds, the global model learns from their data, and their statistical utility decreases. This is a useful property as it changes the order of clients; recently selected clients go down, and new clients with slightly smaller utility per unit of carbon have a higher ranking. This intrinsic shuffling of clients exposes EcoLearn to a broader range of clients, allowing it to select clients with lower carbon.

3 – Impact of time-varying number of clients. FL client selection approaches such as Oort and REFL select the same number of clients across all rounds. However, prior work on ML training in centralized and federated settings has highlighted the presence of an initial period in training when the model learns faster. During this period, an exposure to a large number of high-quality clients can enable it to reach model accuracies that are not possible otherwise [12, 13]. This period is akin to, and gets its name from, the *critical learning period*-(CLP) in humans, where children learn very fast. Figure 7 shows that selecting more clients in the early rounds to exploit the CLP can result in the same accuracy as selecting all clients at all times with astronomical carbon costs. EcoLearn amplifies this effect by selecting high-utility clients during the initial rounds, offsetting the suboptimality of later selection based on utility per unit of carbon.

EcoLearn exploits low carbon periods to over-provision the number of clients in each training round, accommodates stragglers to reduce the training time, while ensuring the carbon cost of resulting wasted computation remains low.

Comparison with related FL frameworks. Table 2 shows that EcoLearn distinguishes itself from other resource-efficient approaches in FL by integrating carbon awareness into multiple aspects of client selection—an aspect not addressed by prior work. The only existing approach that incorporates carbon awareness into client selection is FedZero [24], as it focuses solely on utilizing renewable and spare energy to reduce carbon costs. This work is orthogonal to EcoLearn, as it selects only those clients powered by zero-carbon energy sources (such as excess renewable energy or spare computing capacity). This strategy may result in certain clients being unavailable for extended periods or, alternatively, relying on carbon-intensive grid energy when zero-carbon sources are not accessible—thereby increasing training time and/or the overall carbon footprint. In summary, EcoLearn *selects high-utility but low-carbon clients during the early rounds and opportunistically launches replicas during low-carbon periods to mitigate stragglers.* As shown in Section 4, EcoLearn significantly reduces the carbon footprint while maintaining the accuracy and training time of the carbon-agnostic policies.

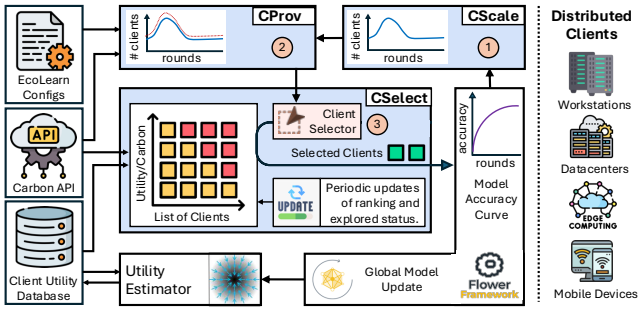


Figure 8: EcoLearn design and its components.

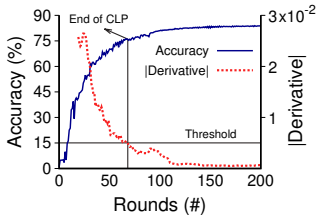


Figure 9: Finding the critical learning period (CLP) over rounds.

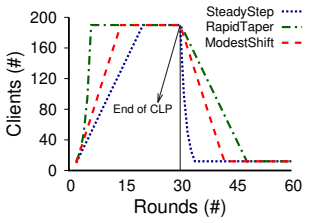


Figure 10: Variants of client scaling strategies in our work.

3 EcoLearn Design

Figure 8 provides an overview of EcoLearn's general design, which includes carbon-aware policies as a per-round client selection mechanism. Our client selection mechanism can easily be extended to different federated learning (FL) frameworks. A design choice enabling integration with higher-level APIs that provide FL-based inference services.

EcoLearn integrates with and extends existing FL frameworks that already provide basic functions of communication, client configurations, and model updates at the aggregator. However, FL frameworks do not provide native support for monitoring carbon footprint, or any other cost, of the clients. We had to implement the auxiliary modules needed for carbon awareness, such as the time- and location-specific carbon information services. As discussed in Section 4.1, we chose the Flower Framework [14] in our implementation, which is one of the most widely used FL frameworks for FL research and practical deployments. However, our design also applies to similar frameworks that enabling configuring the list of clients that are selected in each round of training.

We next detail the EcoLearn modules and describe how they enable EcoLearn to make three decisions:

❶ How many clients should it select in each round to exploit the learning benefit of early rounds? (Section 3.1)

❷ Given the desired number of clients for a round, how many extra clients should it provision to ensure that the desired number of clients respond in time? (Section 3.2)

❸ In selecting a specific number of clients, which specific clients should it select? (Section 3.3)

In detailing these modules, we highlight our design decisions to make these modules carbon-aware and the challenges we faced in practically implementing our solutions.

3.1 CScale: Time-varying Client Scaling

FL typically uses the same number of clients for all training rounds. However, prior work on analyzing learning patterns in ML and FL suggests that selecting more clients in the earlier training rounds is critical for creating strong connections that better understand the distribution of client data [12, 13, 25]. As we discussed in Section 2.3, and shown in Figure 7, the learning boost from a higher number of clients in earlier rounds enables the model to increase the accuracy achieved at the end of training [12, 13]. EcoLearn exploits this phenomenon to reduce the number of rounds and carbon emissions needed to reach a target accuracy.

In practice, *scaling the clients across rounds* to exploit the critical learning period is non-trivial, as when it starts, how long it lasts, and how many clients are enough during this period is not known *a priori*. In addition, the answers to these questions change based on model- and data-specific factors. Therefore, the critical learning period must be identified online in a way that adapts to different FL settings and environments. To solve this problem, EcoLearn develops a mechanism to detect the start and end of the critical learning period and a set of policies that scale the number of clients.

Detecting critical learning period. Prior work analyzes the weighted aggregate of loss gradients across all the clients during their local training to identify when the learning slows down [12, 13]. However, this approach is intrusive and incurs the overhead of collecting gradients. To tackle this challenge, as stated in Algorithm 1, CScale uses the accuracy curve of the global model to mark the start and end of CLP (line 1). Figure 9 shows the curves for both the accuracy (left y-axis) and the magnitude of its derivative (right y-axis). Typically, accuracy increases sharply during the initial learning phase, resulting in a high, rapidly increasing derivative, indicating the CLP. Gradually, the slope of the accuracy curve decreases, and its derivative starts approaching 0. As the magnitude of the derivative falls below a predefined threshold (τ_{deriv}) set by EcoLearn, the detection algorithm marks that point as the conclusion of the CLP.

Scaling the number of clients. Once the CLP starts, the ClientScaler scales the number of clients until the magnitude of the derivative falls below the specified threshold; at which point it starts decreasing the number of clients as mentioned at line 4. CScale enables three scaling policies with different scaling up and down factors. Adjusting these factors affects the training performance and carbon cost.

Algorithm 1: `ClientScaler`

Input : Scaling policy ($\mathcal{P}$), CLP detection threshold (τ_{deriv}), minimum no. of clients ($N_{\min}$)
Output : N_{CLP}

A ← `AccuracyCurve()` A'(r) ← $\frac{dA}{dr}$;
$\mu \leftarrow |A'(r)|$;
$\mu_r \leftarrow \frac{1}{v}\sum_{i=r-v+1}^{r}|A'(i)|$;
$N_{\text{CLP}} \leftarrow$ `IncreaseClients`($\mathcal{P}$) **if** $\mu_r <$ τ_{deriv} **else** max(`DecreaseClients`($\mathcal{P}$), $N_{\min}$)
return N_{CLP}

Algorithm 2: `StragglerProvisioning`

Input : N_{CLP}, carbon weight (ω_{carbon}), training time weight (ω_{time})
Output : CLP and staggler-aware (SA) number of clients $N_{\text{CLP-SA}}$

over_prov_factor ← `CarbonTrainTimeTradeoff`(ω_{carbon}, ω_{time})
$N_{\text{CLP-SA}} \leftarrow N_{\text{CLP}} \times$ (1 + over_prov_factor)
return $N_{\text{CLP-SA}}$

(1) **`SteadyStep`** – Clients are added steadily during CLP and dropped steeply after CLP. Each round adds $\alpha_1 \times N$ clients (α_1 is a small value). Post CLP, $\beta_1 \times N$ clients are dropped (β_1 is relatively larger).
(2) **`RapidTaper`** – Clients are added rapidly during CLP and dropped steadily after CLP. The values of α_2 and β_2 are reversed, with α_2 being large and β_2 being small.
(3) **`ModestShift`** – Clients are added and dropped at a moderate rate, with α_3 and β_3 being similar values.

Here, $\alpha_2 > \alpha_3 > \alpha_1$, conversely, $\beta_1 > \beta_3 > \beta_2$ and N is the total number of clients. Figure 10 illustrates how the number of clients is scaled up and down based on these strategies. Note that `SteadyStep` minimizes the time during training when a higher number of clients are selected and thus reduces the carbon cost the most while maintaining the accuracy compared to the baseline policies. We show the effect of scaling factors on performance in Section 4.4.

3.2 `CProv`: Straggler-Aware Client Provisioning

The client scaling component discussed in the previous section aims to reduce the average number of clients per round, lowering carbon costs. However, selecting a smaller number of clients may lead to an unexpected drop in accuracy and an increase in training time if some of the selected clients are stragglers. Recall from Section 2.3 that stragglers take longer to complete their training than non-stragglers, and thus, they can potentially slow down the overall FL training process. Thus, `EcoLearn`'s second design component focuses on determining *how many extra* clients we need to provision to mitigate the effect of stragglers.

Existing straggler mitigation techniques use a static approach, over-provisioning a fixed number of clients per training round [4, 15, 24]. This over-provisioning creates a trade-off between carbon cost and training time: increasing replication reduces training time but raises carbon costs. However, beyond a certain threshold, adding clients yields diminishing returns in time reduction, severely limiting carbon savings. `EcoLearn` addresses this trade-off by adjusting the replication rate based on the carbon cost distribution across clients and over time (Section 4.3). When the carbon cost is high, `EcoLearn` selects fewer extra clients, and vice versa.

Our approach, outlined in Algorithm 2, uses the `CarbonTrainTimeTradeoff()` function to mitigate stragglers by balancing training time against carbon cost. This decision is governed by two policy weights: ω_{carbon} and ω_{time}. These weights determine the client replication rate (the number of extra clients provisioned to avoid-stragglers). A higher ω_{carbon} value directs the function to prioritize carbon savings by selecting fewer extra clients (low replication rate), while a higher ω_{time} value prioritizes speed by selecting more clients (high replication rate). The outcome of this process is a carbon-aware selection of clients that optimizes the learning period while mitigating stragglers.

3.3 `CSelect`: Carbon-Aware Client Selection

The carbon-aware client selection component is a fundamental building block of `EcoLearn`, which aims to select clients with the lowest carbon cost and highest utility to achieve high model accuracy. In designing `EcoLearn`'s client selection policy, the carbon cost is based on computing's carbon intensity ($g \cdot CO_2eq/cycle$):

$$\text{CI}_{\text{comp}} = \text{CI}_{\text{energy}}/\text{EE}_{\text{comp}}$$

where $\text{CI}_{\text{energy}}$ is the carbon intensity of energy ($g \cdot CO_2eq/kWh$) and EE_{comp} represents a client's computational energy efficiency ($cycle/kWh$). Computing's carbon intensity depends on both energy's carbon intensity and each client's energy-efficiency. `EcoLearn` estimates the current carbon cost when determining which clients to select, which naturally considers the variations in carbon-intensity over time. We devise two baseline carbon-aware selection policies:
1 – Carbon-based (`carbon`). A naive client selection policy selects $\mathcal{K}$ clients in each round with the lowest carbon cost. While this policy provides the maximum carbon savings, it ignores clients' utility in improving the model performance during client selection, significantly increasing training time and lowering accuracy due to insufficient data coverage.
2 – Carbon- and utility-based (`carbon+utility`). The carbon-based approach effectively reduces the overall carbon footprint of training; however, selecting clients solely based

Algorithm 3: `ClientSelection`

Input : CLP and straggler-aware number of client (N_{CLP-SA}), exploration factor (ϵ), carbon weight (ω_{carbon}), utility weight (ω_{util})
Output : Final set of client S

1 $\lambda \leftarrow$ `UtilityCarbonRanking`(ω_{carbon}, ω_{util})
2 $S_{explore} \leftarrow$ `GetUnexploredClients`($\epsilon \cdot N_{CLP-SA}$, λ)
3 $S_{exploit} \leftarrow$ `GetExploitationClients`($(1-\epsilon) \cdot N_{CLP-SA}$, , λ)
4 $S \leftarrow S_{explore} + S_{exploit}$
5 **return** S

on carbon can introduce bias [26] which impacts the model accuracy. Instead, we select clients based on their normalized utility [4] per unit carbon in each round. We call this policy a "`utility+carbon`" aware client selection policy.

For a given client k, we define this efficiency metric as:

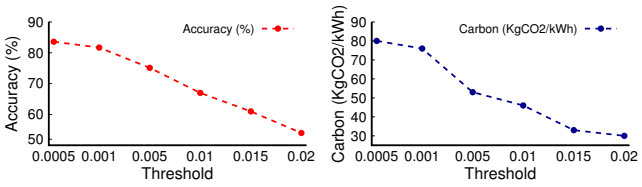
$$E_k = \frac{U_k}{C_k} = \frac{|D_k|}{C'_k} \times \sqrt{\frac{1}{|D_k|} \sum_{i \in D_k} \text{Loss}(i)^2}.$$

where, U_k is the statistical utility, and C'_k represents the instantaneous computing's carbon intensity associated with client k at location x. $|D_k|$ is the total number of training samples from client k, and $\text{Loss}(i)$ is the training loss for sample i. Clients with higher accumulated losses are considered more significant for future rounds.

This policy outperforms the accuracy of a simple `carbon`-based approach by incorporating utility alongside carbon cost when selecting clients. While it effectively reduces the overall carbon footprint of FL training, the potential exclusion of high-utility, high-carbon clients limits model accuracy compared to carbon-agnostic utility-based approaches, which may represent an acceptable but undesirable trade-off. Thus, making existing carbon-agnostic utility-based techniques carbon-aware is insufficient, as it significantly reduces accuracy (shown in Section 4.2). This limitation motivates `EcoLearn` 's additional optimizations.

3.4 `EcoLearn`'s Policy

`EcoLearn`'s policy combines the functionality of all the previous modules, as described in Algorithm 4. At the start of training, `EcoLearn`'s client selection policy initiates by selecting clients in the first round to calculate utility values, which are then stored in a client utility database. The process begins by selecting $M \times N$ clients, where M is a fraction in the range $[0, 1]$, and N denotes the total number of clients. In subsequent rounds, `EcoLearn` receives a list of clients (λ) and ranks them based on E_k Algorithm 3-line 1. Since the utility U_k is determined only after a client participates in training, sampling all clients individually becomes impractical for large numbers. To address this, `EcoLearn` uses an



(a) Accuracy (b) Carbon Cost

Figure 11: Effect of varying detection threshold.

exploration-exploitation technique, balancing between utilizing clients with known utilities (exploitation) and exploring new clients to identify those with high utility (exploration). The system selects (exploits) $(1-\epsilon) \times N_{CLP-SA}$ clients from the already explored clients in the λ list (line 3), while the remaining $\epsilon \times N_{CLP-SA}$ clients are selected from the unexplored client set (line 2). Here, ϵ takes values in the range $[0, 1]$, controlling the rate of exploration.

Simultaneously, `CScale` is monitoring for the start of CLP. If the CLP has not yet started, it returns a default value for the number of clients (i.e., $M \times N$). Once CLP starts, it uses the scaling policies to determine the number of clients for the next round. In either scenario, the number of clients provided by the `CScale` module is passed to the `CProv` module, which adds the extra number of clients based on the value it received (N_{CLP}) and the average carbon intensity across clients. The final number of clients is then passed to the `CSelect` module, which selects specific clients based on their normalized utility per carbon values while accounting for the need to explore new clients for utility estimation. As discussed in Section 2.3, the utility of the clients that have already participated in the training process decreases over time, and the unexplored clients are highly likely to have higher utility, which helps with selecting high-value clients at the start. This also ensures that `EcoLearn` does not repeatedly select the same clients during the training process, which could introduce bias and negatively impact model performance. For each round, we get the final set of clients that should participate in the training process. These rounds continue until the global model reaches the training time threshold or hits the test target accuracy.

4 `EcoLearn` Evaluation

We next present our evaluation methodology (Section 4.1), evaluate `EcoLearn`'s three core design components (Section 4.2 to Section 4.4), and analyze `EcoLearn`'s generalizability across real-world datasets (Section 4.5).

4.1 Evaluation Methodology

We present our evaluation methodology, covering (1) implementation and compute resources, (2) training datasets and models, (3) carbon cost traits, and (4) baseline policies.

Algorithm 4: EcoLearn Framework

Input: clp policy $\mathcal{P}$, exploration factor ϵ, stopping criteria Λ, CLP threshold τ_{deriv}, carbon cost weight ω_{carbon}, training time weight ω_{time}, utility weight ω_{util}, minimum clients N_{min}
Output: final model parameters Θ_{final}

```
while Λ ≠ true do
    N_CLP ← ClientScaling(P, τ_deriv, N_min)
    N_CLP-SA ← StragglerProv(N_CLP, ω_carbon, ω_time)
    S ← ClientSelection(N_CLP-SA, ω_carbon, ω_util, ϵ)
    StartTraining(clients_round)
    n ← 0
    while n < N_CLP do
        n ← ClientsFinishTraining()
    Θ_final ← UpdateModelParams(n)
return Θ_final
```

Implementation. Our implementation extends the Flower framework [14], which emulates the communication between a global server and geographically distributed clients. We use PyTorch as the underlying ML library within Flower. We leverage Flower's Virtual Client Engine (VCE) to configure and manage resources, enabling efficient FL execution. The virtual clients are implemented using Ray [27].

We implement a custom aggregation strategy within the Flower framework to support carbon- and utility-aware client selection, carbon-aware straggler provisioning, and time-varying client scaling, based on EcoLearn 's design (Section 3). Specifically, we design a custom `select_clients()` method that selects clients by jointly optimizing statistical utility and carbon cost in each training round. To support straggler mitigation, we implement a custom `fit_clients()` method within the server, which enables over-provisioning by launching training with additional clients. This method tracks responses and terminates the round as soon as the target number of updates is received. Our strategy also extends `configure_fit()` to integrate adaptive scaling based on carbon load phases (CLP). During CLP, the number of clients is scaled up to exploit low-carbon availability and scaled down when the CLP ends. This behavior is governed by a policy defined through tunable scaling parameters, as detailed in Section 3.1. Our implementation adds approximately 2000 lines of code.

Experimental setup. We evaluate EcoLearn on a cluster of 40 `NVIDIA GTX 1080Ti` and 6 `NVIDIA RTX 8000` GPUs to emulate a medium-to-large-scale FL setup. We emulate up to 1,000 clients in the FL training, representing large-scale edge deployments using Flower [14]. Our emulation and evaluation setup is comparable to or larger than that used in prior work on optimizing FL training [4, 5, 13, 24].

Datasets and models. We use three datasets in our experiments, each with distinct scale and complexity: EMNIST [28], Shakespeare [29], and Google Speech Commands [30]. Table 3 summarizes their characteristics. EMNIST, a vision dataset for image recognition, Shakespeare contains dialogues for next-character prediction, and Google Speech Commands for speech recognition tasks.

Table 3: Statistics of datasets used in the evaluation.

Datasets Datasets	Samples	Emulated Clients (N)	Inherently non-iid
EMNIST	814,255	1,000	No
Shakespeare	422,615	660	Yes
Google Speech	105,829	100	Yes

Inducing stragglers. Recalling Section 2.3, stragglers impact FL's efficiency by increasing training time. To enable realistic FL evaluation and simulate real-world delays, we induce stragglers in our system using heavy-tailed distributions [31], meaning some clients fall in the heavy tail and experience more delay than others.

Achieving non-iid data distributions. We leverage the Fang distribution [32] to precisely control and quantify the impact of data heterogeneity (non-iid *ness*). The level of "non-iid *ness*" is controlled using a parameter, κ, which ranges from 0 (fully i.i.d) to 1 (fully non-iid). At $\kappa = 1$, a single client contains only one class with no representation of other classes. At $\kappa = 0$, all clients hold local data of all classes in equal proportions. For $0 < \kappa < 1$, the degree of non-iid *ness* ranges from mild to extreme non-iid.

Detection threshold (τ_{deriv}) in CLP. As mentioned in Section 3.1, if the magnitude of the derivative falls below a threshold (τ_{deriv}), EcoLearn marks that point as the conclusion of CLP. Changing the value of this threshold impacts both the accuracy and the carbon cost of training. As τ_{deriv} increases, CLP consists of fewer rounds, causing it to shrink. Consequently, both accuracy and carbon footprint decrease, as EcoLearn utilizes a larger number of clients in fewer rounds. Conversely, a smaller τ_{deriv} results in better accuracy at the cost of higher carbon emissions. We provide a sensitivity analysis for this threshold in Figure 11, which informed our choice of $\tau_{\text{deriv}} = 0.005$ from a tuned range of $0.0005 < \tau_{\text{deriv}} < 0.03$. The effectiveness of the EcoLearn, which incorporates our CLP detection method, is demonstrated in an end-to-end comparison against the `criticalFL` baseline in Section 4.4, Figure 18 and Figure 19.

Client scaling factors. In Section 3.1, we discussed three client scaling policies that vary the number of clients during and after CLP. We follow the pattern $\alpha_2 > \alpha_3 > \alpha_1$ and $\beta_1 > \beta_3 > \beta_2$, setting the scaling factor values as: $\alpha_1 = 0.01$, $\alpha_2 = 2$, $\alpha_3 = 0.015$, and $\beta_1 = 2$, $\beta_2 = 0.01$, $\beta_3 = 0.015$. These scaling factors (inspired but distinct from prior work [12, 13])

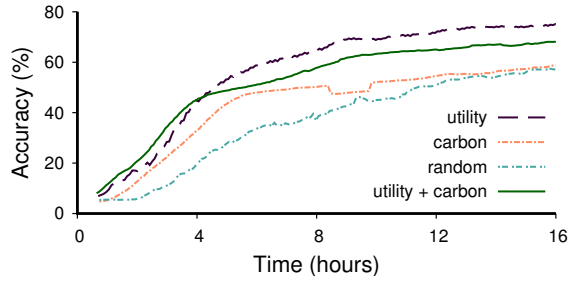


Figure 12: Accuracy comparison of baseline against EcoLearn's non-CLP policies on the EMNIST dataset (A higher accuracy is better).

simulate different scaling behaviors, including steady scale-up, rapid scale-down, rapid scale-up, steady scale-down, and moderate scale-up and scale-down respectively.

Model parameters. We train ResNet-18 [33] on EMNIST for image classification, a two-layer LSTM [34] on Shakespeare for next character prediction, and KWT-1 [35] on Google Speech for speech classification. Initial learning rates were 0.01 for EMNIST, 0.001 [35] for Google Speech, and 0.8 [34] for Shakespeare, with batch sizes of 20 for EMNIST and Shakespeare, and 16 for Google Speech.

We use FedAvg [2] as the aggregation strategy across all datasets and selection policies as it is the commonly used in practice and in baseline models. Other techniques like SCAFFOLD [36] or FedProx [34] can be used in future work.

Baseline policies. Below, we present the baseline approaches that we compare EcoLearn with.

(1) **Random selection.** Random client selection [15, 19] ensures fairness and reduces bias. However, it extends training time by including clients with less value or noisy data, leading to slower learning and more rounds to reach the desired accuracy.

(2) **Guided selection - Oort.** As mentioned in Section 2.2 Oort [4] employs a guided participant selection approach to prioritize clients with higher statistical utility to maximize system efficiency. This approach enhances time-to-accuracy by utilizing client data, significantly improving model performance and outperforming random selection.

To mitigate stragglers' induced delays, all the policies shown in Figure 12 to Figure 14 over-provision clients using a $1.3\mathcal{K}$ rule. Thus, provisioning 30% more clients than the $\mathcal{K}$ required, training all $1.3\mathcal{K}$ clients, and collecting updates from $\mathcal{K}$ clients that respond the fastest.

Evaluation Metric. To evaluate the performance and carbon footprint of carbon-aware and carbon-agnostic selection policies, we train them for the same duration and compare their accuracy and carbon cost. This approach offers a better evaluation metric than the time to reach a target accuracy, as it avoids infinite carbon costs and training times for policies

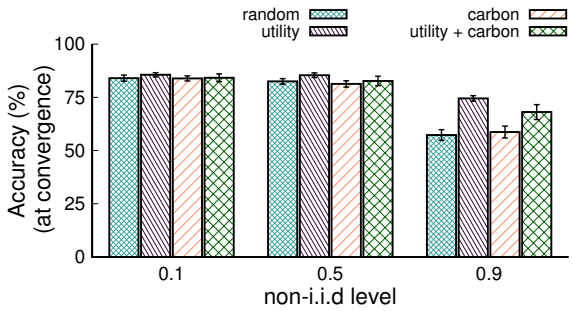


Figure 13: Effect of non-iid levels on the performance of the global FL model under different client selection policies (Higher accuracy is better).

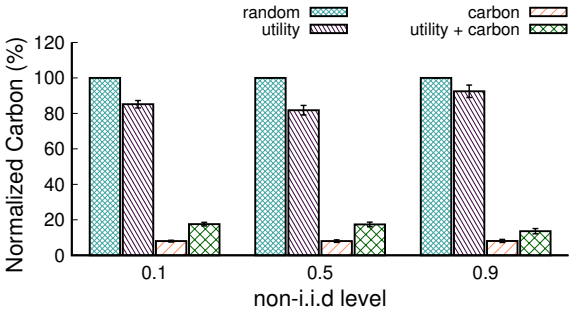


Figure 14: Carbon costs for client selection policies, plotted in comparison with the random baseline. (A lower percentage value is better.)

that never reach the target. We use the utility-based baseline as a benchmark, training all policies for a time T_u, the convergence time for the utility-based approach. We report results across 10 runs of each experiment.

4.2 Carbon-Aware Client Selection

Here, we evaluate the performance and carbon cost of carbon-aware (carbon- and utility+carbon based) policies against carbon-agnostic (random and utility-based) baselines. These policies exclude the CLP and carbon-aware straggler mitigation, discussed in our final policy in Section 4.4.

Accuracy. Figure 12 shows the accuracy for the above-discussed selection policies at a non-iid level of 0.9, with the x-axis representing wall-clock time (in hours). All policies are trained for the same duration—16 hours, equivalent to the utility-based selection's convergence time. Baseline strategies—utility-based and random selection—are shown by dashed magenta and dotted blue lines. Carbon-aware policies, carbon-based and utility+carbon, are represented by dotted orange and solid green lines, respectively.

At 0.9 non-iid, Figure 13 shows that the carbon-aware utility-based selection policy, i.e., utility+carbon, outperforms both random and carbon-based approaches by

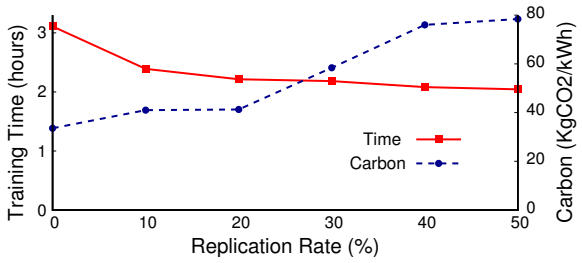


Figure 15: Impact of client replication on training time and carbon cost for straggler mitigation.

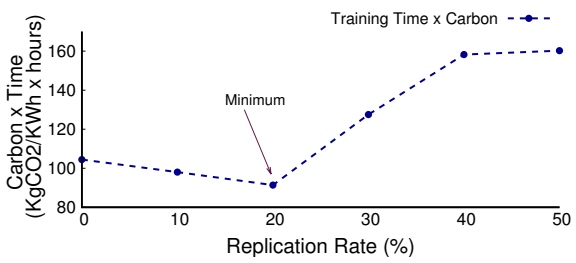


Figure 16: Carbon×time minima represents the optimal trade-off between carbon and training time across different replication rates.

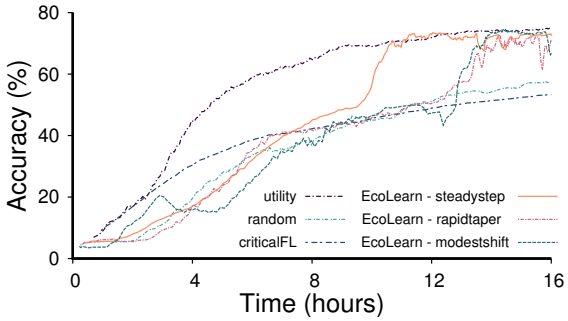


Figure 17: Accuracy comparison of EcoLearn's scaling strategies against the baselines.

selecting high-utility, low-carbon-cost clients. The utility-based selection achieves the highest accuracy, surpassing `utility+carbon` by *6.4%*, and both random and carbon-based approaches by approximately *17%*. The `utility+carbon` policy achieves approximately ***10%*** higher accuracy than the carbon-based and random baselines. This result demonstrates that incorporating both utility and carbon cost into the client selection process leads to better model performance than random selection alone.

However, as discussed in Section 3.3, it is still less effective than the purely utility-based selection, where the introduction of carbon awareness reduces accuracy (by *6.4%* in this case). This degradation stems from the trade-off between maximizing utility (which improves performance) and minimizing carbon cost, where clients with lower carbon costs may not necessarily have high utility, thus yielding lower model accuracy. As we show in Section 4.4, EcoLearn leverages this insight by selecting clients with higher utility per unit of carbon during the CLP, effectively narrowing the accuracy gap between `utility+carbon` and utility-based selection, and achieving performance comparable to the latter.

Carbon cost. Figure 14, with a non-iid level of 0.9, shows the carbon cost for training time equivalent to the utility-based selection's convergence time. The y-axis shows the carbon costs of client selection policies in comparison to random selection. The `utility+carbon` outperforms carbon-agnostic policies, reducing carbon consumption by ***86.4%*** compared to random and by ***85.3%*** compared to utility-based selection. While carbon based approach emits the least carbon, it suffers a significant performance loss.

Effect of data heterogeneity. Figure 13 shows the effect of varying non-iid levels on client selection policies. At low non-iid values (0.1), the final model accuracy is similar across all selection policies, as the data is uniformly distributed among all clients, leading to similar contributions towards the model's performance. As the non-iid level increases to 0.5, differences in accuracy become more pronounced. The utility-based approach performs best in terms of accuracy by prioritizing clients with higher statistical utility, due to the pronounced effect of non-iid data across clients and some clients contributing more towards the performance of the model than others. However, in terms of carbon savings, the carbon-aware selection policies perform better as they prioritize selecting clients based on carbon, giving more weight to the carbon savings than to the accuracy. The carbon based approach reduces carbon cost the most (*92% less than the random baseline*). The `utility+carbon` approach achieves substantial savings of ≈*79%* and *82.6%* compared to utility-based and random-based, respectively.

At non-iid level of 0.9, the effect of non-iid data across clients in-terms of clients' utility is more prominent. We observe that `utility+carbon` performs ***10%*** better than random and carbon based selections in accuracy, while giving *6.4%* less accuracy than utility-based selection. Additionally, Figure 14 shows that `utility+carbon` reduces carbon consumption by ***86.4%*** compared to random and ***85.3%*** compared to utility-based selection.

Key point. *Our results show that* `utility+carbon` *policy of* EcoLearn *consistently outperforms random and utility-based selection baselines on carbon cost across all the levels of non-*iid *while maintaining the accuracy of the model.*

4.3 Straggler-Aware Client Provisioning

Recalling Section 3.2, we over-provision clients in each training round to mitigate the effect of stragglers on training time. Our design navigates the trade-off between wasted carbon

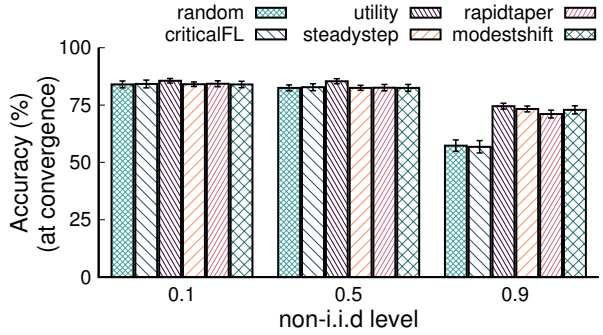


Figure 18: Effect of non-iid levels on the accuracy of EcoLearn's scaling policies, compared to baselines.

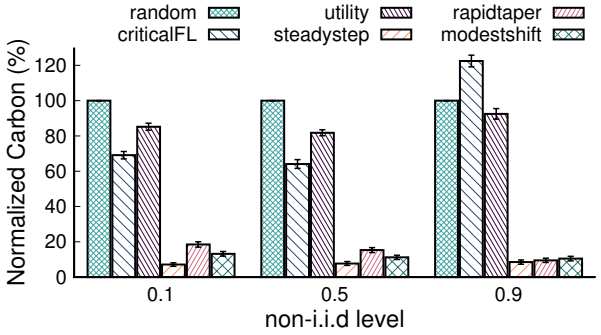


Figure 19: Carbon cost of EcoLearn's scaling policies at different non-iid levels, compared to baselines.

cost due to stragglers and reduced training time due to straggler mitigation. To analyze this trade-off, we compare the behavior of training time and carbon cost for FL at different levels of replication, aiming to mitigate delays induced by stragglers, as shown in Figure 15. The x-axis represents the replication rate, the left y-axis shows the total training time, and the right y-axis illustrates the carbon cost of training across various replication rates. At 0% replication, the training time is large due to the presence of stragglers, but carbon cost remains low, since no extra clients are over-provisioned. Higher replication rates (50%) reduce training time by *33%* but increase carbon cost by over *100%*. Beyond a certain replication rate, training time decreases marginally due to the heavy-tailed distribution of stragglers.

Based on these observations, we optimize the carbon-time metric with respect to replication rate, indicating there is a specific replication rate that maximizes the benefit of reducing carbon cost relative to the overall training time. Figure 16 shows the relationship between carbon-time and the replication rate, highlighting a minimum at ***20% replication rate*** (equivalent to $1.2\mathcal{K}$). This replication rate provides the most effective balance between reducing carbon cost and improving the training time. *Our evaluation demonstrates that the optimal replication rate is strongly influenced by the clients' carbon cost distribution.* Thus optimizing the replication rate based on the carbon cost distribution can yield significant improvements in reducing the carbon cost and the training time, than to use a static $1.3\mathcal{K}$ over-provisioning factor.

4.4 Time-Varying Client Scaling

Recalling Section 4.2, `utility+carbon` outperforms random selection but is less effective than utility-based selection due to its trade-off between utility and carbon cost, leading to client selection that reduces carbon cost but does not maximize accuracy. In this section, we present that by varying the number of clients within and after CLP, we can increase the accuracy and further reduce the carbon cost. Additionally, we introduce another baseline approach, criticalFL [13], which dynamically detects CLP in the FL training process and adaptively determines the number of clients to participate in each FL training round. In contrast to EcoLearn's selection policy, which selects clients based on their utility and carbon cost in each round, criticalFL chooses clients randomly, without considering their utility or carbon cost.

Evaluating EcoLearn's client selection policy. Based on the insights about the carbon-aware straggler mitigation technique discussed in Section 4.3, we adopt a 1.2K replication rate (provisioning 20% more clients and collecting results from the first K clients) for EcoLearn's CLP-based policies. This configuration avoids stragglers during FL training and optimizes the associated training time and carbon cost. In Figure 17, we compare EcoLearn's CLP-based client selection policies to the carbon-agnostic baseline policies. The y-axis represents the performance of all the selection policies at a non-iid level of 0.9, when trained for the same duration—16 hours, equivalent to the utility-based selection's convergence time. This avoids infinite carbon emissions and training times for policies that never reach benchmark accuracy. We observe that the accuracy of EcoLearn's CLP-based client selection policy (*SteadyStep*) reaches ***73.3%***, which is an improvement of *3.5%* compared to the non-CLP version of this policy (`utility+carbon` technique, discussed in Section 4.2). EcoLearn's selection policy is also comparable to utility-based selection (*74.5%*) and performs ***11.2%*** and ***16.5%*** better than random and criticalFL, respectively. These results are shown in Figure 18 at a non-iid level of 0.9.

Figure 17 illustrates an increase in accuracy when the critical learning period ends after around 9 hours of training, with the slope of this increase depending on the decaying factor by which we reduce the number of clients once the CLP ends. This behavior likely arises due to a very high non-iid level and the abrupt change in the number of clients from a large to a small number after CLP. Intuitively, learning becomes more challenging when many clients are provisioned within CLP and becomes easier once CLP ends, as there are

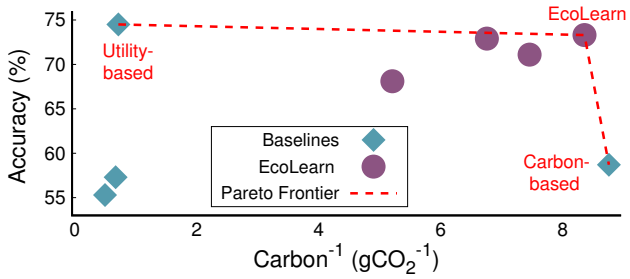


Figure 20: Pareto-frontier of carbon cost against the accuracy. Circles represent EcoLearn's selection policies and stars indicate baselines.

fewer clients with fewer class representations, thus boosting accuracy. This behavior has been reported in [37] and [38].

Figure 19 shows that, at a non-iid level of 0.9, EcoLearn's *SteadyStep* policy reduces carbon cost by ***91.5%***, ***90.8%***, and ***93%*** compared to the random baseline, utility-based selection, and criticalFL, respectively, while maintaining comparable accuracy. Additionally, EcoLearn's *SteadyStep* selection policy, integrated with CLP and carbon-aware straggler mitigation, reduces carbon cost by ***37.5%*** and increases accuracy by ***5.2%*** compared to its non-CLP version (utility+carbon) when trained for the same duration.

Effect of scaling strategies in CLP. Figure 18 shows, EcoLearn's CLP based policies perform comparably in terms of accuracy across all levels of non-iid. However, Figure 19 shows that the EcoLearn's CLP based policies significantly reduce the carbon cost of training across all non-iid levels compared to the baselines.

Among EcoLearn's CLP based policies, *SteadyStep* achieves the greatest reduction in carbon cost by minimizing the duration for which a large client base is active during the training period. As illustrated in Figure 10, *SteadyStep* strategy gradually increases the number of client during CLP and quickly drops them afterward, significantly reducing the carbon costs. In contrast, *RapidTaper* incurs higher carbon costs due to its rapid increase in the number of clients during CLP and maintaining a large client base for an extended period. The *ModestShift* strategy takes a moderate approach in client ramp-up during CLP and reduction post-CLP, resulting in higher carbon costs compared to *SteadyStep*, but lower or comparable costs to *RapidTaper* across all non-iid levels.

In iid settings, where client utility is uniform, EcoLearn's selection policy naturally defaults to prioritizing carbon cost. Consequently, in near-iid conditions (non-iid level = 0.1), EcoLearn's *SteadyStep* carbon footprint, shown in Figure 19, is nearly identical to that of a purely carbon-aware policy, as seen in Figure 14. This inherent efficiency allows EcoLearn's *SteadyStep* policy to significantly outperform other approaches, including *RapidTaper*, *ModestShift*, and other non-CLP baselines. When trained for the same duration, *SteadyStep* achieves a carbon cost that is **10.8**×, **11.7**×, and **14.4**× lower than random selection, utility-based selection, and criticalFL, respectively.

Pareto-frontier. We can present our problem as a multi-objective optimization problem between carbon cost and accuracy, in the form of a Pareto frontier. As we know, the Pareto frontier is a set of solutions that represents the best trade-off between all the objective functions, we can use this information to represent the best trade-off between the carbon cost and accuracy across multiple clients selection policies. Figure 20 illustrates the Pareto frontier for the trade-off between accuracy (y-axis) and the inverse of carbon cost (x-axis). We aim to maximize both our objective functions as we want to determine which policies lie on the Pareto frontier by maximizing the accuracy and the carbon reductions. Figure 20 shows, EcoLearn resides on the Pareto frontier, signifying its optimal balance between accuracy and carbon relative to baseline client selection approaches.

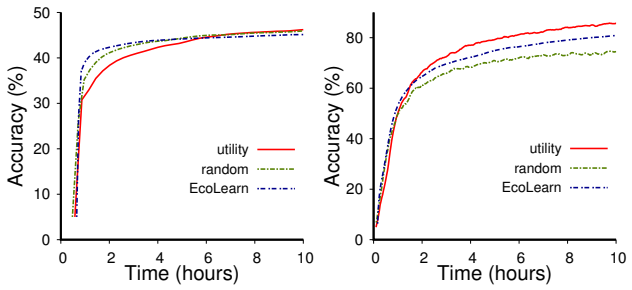
Key Point: EcoLearn's *client selection policy (SteadyStep), integrated with CLP and a carbon-aware straggler mitigation technique, outperforms all baseline selection policies on carbon cost across all non-iid levels. It matches or surpasses them in accuracy while maintaining consistent training times.*

4.5 EcoLearn on Real-World Datasets

In the Section 4.2 to Section 4.4, we present the evaluation results for the EMNIST dataset. Here, we evaluate EcoLearn with two additional, distinct ML datasets to demonstrate its generalizability. We use the Shakespeare and the Google Speech dataset, both are inherently non-iid, i.e., data heterogeneous. We compare EcoLearn's selection policy (*SteadyStep*) to random and utility-based selection.

Shakespeare. Figure 21(a) illustrates the accuracy of baseline selection policies and EcoLearn's selection policy on the y-axis against wall-clock time (hours) on the x-axis. We observe similar accuracy across all three selection strategies after training for the same duration, with utility-based selection performing slightly better. This behavior aligns with expectations and findings from recent studies [24, 39]. Table 4 shows that EcoLearn achieves a ***76.5%*** and ***73.1%*** reduction in carbon cost compared to random selection and utility-based selection when trained for the same duration.

Google Speech. Figure 21(b) shows the accuracy of baseline selection policies (random- and utility-based) and EcoLearn's selection policy on the y-axis against wall-clock time (hours) on the x-axis. When trained for the same duration, EcoLearn achieves an accuracy of *80.7%*, and utility-based selection achieves *85.8%*. EcoLearn performs better than the random baseline which achieves *74.5%* accuracy, as



(a) Shakespeare (b) Google Speech

Figure 21: Accuracy comparison of EcoLearn against the random and utility-based baselines when trained on Shakespeare and Google Speech datasets.

shown in Table 4. While we sacrifice accuracy, EcoLearn reduces carbon cost by a huge margin, surpassing both random and utility-based client selection approaches with carbon reductions up to ***65.7%*** and ***68%***, respectively. This shows the fundamental trade-off between accuracy and carbon cost.

Key Point: EcoLearn *outperforms the carbon-agnostic baselines across different ML models and datasets suggesting that its carbon cost and accuracy tradeoffs are generalizable.*

5 Related Work

Data distribution. FL uses privacy-preserving distributed algorithms [40, 41], keeping data on clients' premises, which may be geographically diverse [42]. Unlike traditional ML, which assumes i.i.d data [43], FL handles data from disparate probability distributions. We explore FL's non-i.i.d challenges [44] in relationship with carbon, utility, and CLP.

Client selection strategies [45, 46]. In FL, a key challenge is crafting a robust model from a large pool of clients while minimizing computation and communication overhead. Various methodologies have emerged to enhance model performance [47], optimize communication cost with the FL aggregator [2, 12, 34, 48, 49], and reduce the computational load on the clients [47, 50]. However, identifying clients whose contributions enhance the global model's accuracy significantly remains underexplored. Recent works have introduced a metric of statistical utility [4, 51, 52], which we leverage in EcoLearn. Others have focused on reducing the number of clients per round while maintaining convergence rates [53, 54]. Additionally, FLAME framework [55] assesses clients' energy profiles and simulates various energy scenarios across the federation. However, none of these works have addressed the carbon cost or the impact of diverse selection strategies during the training process.

Carbon-aware client selection Recent work [56, 57] explores carbon cost in FL, focusing on measuring the overall carbon footprint rather than optimizing the carbon footprint of operation. Another study [1] optimizes AI model

Table 4: Different selection policies across datasets.

Dataset	Selection Policies	Convergence Accuracy	Carbon Emissions (%)
Shakespeare	Utility based	46.2	87.6
	Random	45.9	100
	EcoLearn	45.1	***23.5***
Google Speech	Utility based	85.8	107.3
	Random	74.5	100
	EcoLearn	80.7	***34.3***

training across geo-distributed data centers by balancing learning performance and carbon emissions within a fixed carbon footprint budget. In contrast, our approach focuses on dynamically reducing carbon emissions by varying the AI workload (varying number of selected clients for training), and we can achieve better overall performance and greater carbon reduction. Additionally, FedZero [24] uses renewable resources to reduce carbon costs. This work is orthogonal to EcoLearn, as it has different assumptions and constraints. In particular, FedZero only selects clients with zero-carbon energy, which may result in some clients being unavailable for extended periods, thereby increasing the training time.

Critical learning period Recent studies highlight the Critical Learning Period (CLP) in shaping DNN model quality in centralized ML [58, 59]. Insufficient data during CLP can cause permanent model degradation. This concept extends to FL, where early learning phases can significantly influence the final accuracy [12, 13] and increase vulnerability to adversarial attacks [60]. EcoLearn leverages CLP insights to explore optimal client addition or removal rates during and after CLP for better performance.

6 Conclusion

We introduce EcoLearn, which leverages clients' varying carbon cost. EcoLearn selects clients based on their carbon cost and statistical utility to reduce carbon emissions, while maintaining accuracy. Additionally, our investigation into critical learning periods inspired dynamic client selection, and scaling of clients to optimize cost without a significant impact on accuracy. To further reduce carbon, we optimize the tradeoff between wasted carbon due to stragglers and reduced training time due to straggler mitigation techniques. Our evaluation shows that EcoLearn reduces the carbon footprint of FL training (by up to 10.8×) while maintaining model accuracy and training time (within ~1%) compared to state-of-the-art approaches.

Acknowledgments. This work is partially supported by the National Science Foundation (NSF) through grants 2213636, 2325956, 2211888, 1925464, and 1925658. Additionally, it is partially supported by the European Union's Horizon Europe project under grant agreements No 101192749.

References

[1] J. Bian, L. Wang, S. Ren, and J. Xu, "CAFE: Carbon-Aware Federated Learning in Geographically Distributed Data Centers," in *Proceedings of International Conference on Future and Sustainable Energy Systems*, 2024.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of Artificial Intelligence and Statistics*, 2017.

[3] European Parliament and Council of the European Union. (2016) Regulation (EU) 2016/679 of the European Parliament and of the Council. [Online]. Available: https://data.europa.eu/eli/reg/2016/679/oj

[4] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient Federated Learning via Guided Participant Selection," in *Symposium on Operating Systems Design and Implementation*, 2021.

[5] A. M. Abdelmoniem, A. N. Sahu, M. Canini, and S. A. Fahmy, "REFL: Resource-Efficient Federated Learning," in *Proceedings of Computer Systems*, 2023.

[6] L. Sani, A. Iacob, Z. Cao, B. Marino, Y. Gao, T. Paulik, W. Zhao, W. F. Shen, P. Aleksandrov, and X. Qiu, "The Future of Large Language Model Pre-training is Federated," *arXiv*, 2024.

[7] P. Villalobos, A. Ho, J. Sevilla, T. Besiroglu, L. Heim, and M. Hobbhahn, "Will We Run Out of Data? Limits of LLM Scaling Based on Human-Generated Data," *arXiv*, 2024.

[8] A. Douillard, Q. Feng, A. A. Rusu, R. Chhaparia, Y. Donchev, A. Kuncoro, M. Ranzato, A. Szlam, and J. Shen, "Diloco: Distributed Low-Communication Training of Language Models," *arXiv*, 2023.

[9] L. Sani, A. Iacob, Z. Cao, R. Lee, B. Marino, Y. Gao, D. Cai, Z. Li, W. Zhao, X. Qiu *et al.*, "Photon: Federated LLM Pre-Training," *arXiv*, 2024.

[10] F. Strati, P. Elvinger, T. Kerimoglu, and A. Klimovic, "ML Training with Cloud GPU Shortages: Is Cross-Region the Answer?" in *Machine Learning and Systems*, 2024.

[11] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. Jain, M. Rabbat, and K. Hazelwood, "Sustainable AI: Environmental Implications, Challenges and Opportunities," in *Proceedings of Machine Learning and Systems*, 2022.

[12] S. Agarwal, H. Wang, K. Lee, S. Venkataraman, and D. Papailiopoulos, "Accordion: Adaptive Gradient Communication via Critical Learning Regime Identification," in *Proceedings of Machine Learning and Systems*, 2021.

[13] G. Yan, H. Wang, X. Yuan, and J. Li, "CriticalFL: A Critical Learning Periods Augmented Client Selection Framework for Efficient Federated Learning," in *Proceedings of Knowledge Discovery and Data Mining*, 2023.

[14] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, H. L. Kwing, T. Parcollet, P. P. d. Gusmão, and N. D. Lane, "Flower: A Friendly Federated Learning Research Framework," *arXiv*, 2020.

[15] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards Federated Learning at Scale: System Design," in *Proceedings of Machine Learning and Systems*, 2019.

[16] C. T. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated Learning over Wireless Networks: Convergence Analysis and Resource Allocation," *Transactions on Networking*, 2020.

[17] Google, "Google Data Centers: Efficiency," 2024. [Online]. Available: http://google.com/about/datacenters/efficiency/

[18] T. Li, M. Zaheer, A. Beirami, and V. Smith, "Fair Resource Allocation in Federated Learning," in *Proceedings of Learning Representations*, 2020.

[19] D. Huba, J. Nguyen, K. Malik, R. Zhu, M. Rabbat, A. Yousefpour, C.-J. Wu, H. Zhan, P. Ustinov, H. Srinivas, K. Wang, A. Shoumikhin, J. Min, and M. Malek, "PAPAYA: Practical, Private, and Scalable Federated Learning," in *Proceedings of Machine Learning and Systems*, 2022.

[20] C. Li, X. Zeng, M. Zhang, and Z. Cao, "PyramidFL: A Fine-Grained Client Selection Framework for Efficient Federated Learning," in *Proceedings of Mobile Computing And Networking*, 2022.

[21] E. Maps, "Electricity Map," https://www.electricitymap.org/map, 2022.

[22] Watttime, "Watt-Time," https://watttime.org/, 2024.

[23] G. Cloud, "Carbon Free Energy for Google Cloud Regions," https://cloud.google.com/sustainability/region-carbon, 2021.

[24] P. Wiesner, R. Khalili, D. Grinwald, P. Agrawal, L. Thamsen, and O. Kao, "Fedzero: Leveraging Renewable Excess Energy in Federated Learning," in *Proceedings of Future and Sustainable Energy Systems*, 2024.

[25] A. Achille, M. Rovere, and S. Soatto, "Critical Learning Periods in Deep Networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BkeStsCcKQ

[26] D. Y. Zhang, Z. Kou, and D. Wang, "FairFL: A Fair Federated Learning Approach to Reducing Demographic Bias in Privacy-Sensitive Classification Models," in *Proceedings of Big Data*, 2020.

[27] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Ray: A Distributed Framework for Emerging AI Applications," 2018. [Online]. Available: https://github.com/ray-project/ray

[28] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to Handwritten Letters," in *Neural Networks*, 2017.

[29] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A Benchmark for Federated Settings," *arXiv*, 2018.

[30] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *arXiv*, 2018.

[31] A. Behrouzi-Far and E. Soljanin, "Efficient Replication for Straggler Mitigation in Distributed Computing," *arXiv*, 2020.

[32] M. Fang, X. Cao, J. Jia, and N. Gong, "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning," in *Security Symposium*, 2020.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of Computer Vision and Pattern Recognition*, 2016.

[34] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," *Proceedings of Machine learning and systems*, 2020.

[35] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword Transformer: A Self-Attention Model for Keyword Spotting," *arXiv*, 2021.

[36] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic Controlled Averaging for Federated Learning," in *International conference on machine learning*, 2020.

[37] G. Yan, H. Wang, and J. Li, "Critical Learning Periods in Federated Learning," *arXiv*, 2021.

[38] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't Decay the Learning Rate, Increase the Batch Size," *ICLR*, 2017.

[39] J. Shin, Y. Li, Y. Liu, and S.-J. Lee, "Fedbalancer: Data and pace control for efficient federated learning on heterogeneous clients," in *Proceedings of Mobile Systems, Applications and Services*, 2022.

[40] R. C. Geyer, T. Klein, and M. Nabi, "Differentially Private Federated Learning: A Client-Level Perspective," in *NeurIPS*, 2017.

[41] S. Pichai, "Privacy Should Not Be a Luxury Good," 2019. [Online]. Available: https://www.nytimes.com/2019/05/07/opinion/google-sundar-pichai-privacy.html

[42] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied Federated Learning: Improving Google Keyboard

Query Suggestions," 2018.
[43] M. I. Jordan and T. M. Mitchell, "Machine Learning: Trends, Perspectives, and Prospects," *Science*, 2015.
[44] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The Non-IID Data Quagmire of Decentralized Machine Learning," in *International Conference on Machine Learning*, 2020.
[45] L. Fu, H. Zhang, G. Gao, M. Zhang, and X. Liu, "Client Selection in Federated Learning: Principles, Challenges, and Opportunities," *Internet of Things Journal*, 2023.
[46] G. Dániel Németh, M. Ángel Lozano, N. Quadrianto, and N. Oliver, "A Snapshot of the Frontiers of Client Selection in Federated Learning," *arXiv*, 2022.
[47] S. Caldas, J. Konecny, B. McMahan, and A. Talwalkar, "Expanding the Reach of Federated Learning by Reducing Client Resource Requirements," *arXiv*, 2018.
[48] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *Journal on Selected Areas in Communications*, 2019.
[49] W. Y. B. Lim, N. C. Luong, D. T. Hoang *et al.*, "Federated Learning in Mobile Edge Networks: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, 2020.
[50] Z. Xu, F. Yu, J. Xiong, and X. Chen, "Helios: Heterogeneity-Aware Federated Learning with Dynamically Balanced Collaboration," in *Proceedings of Design Automation*, 2021.
[51] T. B. Johnson and C. Guestrin, "Training Deep Models Faster with Robust, Approximate Importance Sampling," in *NeurIPS*, 2018.
[52] A. Katharopoulos and F. Fleuret, "Not All Samples Are Created Equal: Deep Learning with Importance Sampling," in *ICML*, 2018.
[53] W. Chen, S. Horváth, and P. Richtárik, "Optimal Client Sampling for Federated Learning," *Transactions on Machine Learning Research*, 2022.
[54] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily Aggregated Gradient for Communication-Efficient Distributed Learning," in *Advances in Neural Information Processing Systems*, 2018.
[55] H. Cho, A. Mathur, and F. Kawsar, "FLAME: Federated Learning across Multi-device Environments," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2022.
[56] A. Yousefpour, S. Guo, A. Shenoy, S. Ghosh, P. Stock, K. Maeng, S.-W. Krüger, M. Rabbat, C.-J. Wu, and I. Mironov, "Green Federated Learning," *arXiv*, 2023.
[57] X. Qiu, T. Parcollet, J. Fernandez-Marques, P. P. Gusmao, Y. Gao, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "A First Look into the Carbon Footprint of Federated Learning," *Journal of Machine Learning Research*, 2023.
[58] S. Jastrzebski, Z. Kenton, N. Ballas, A. Fischer, Y. Bengio, and A. J. Storkey, "On the Relation Between the Sharpest Directions of DNN Loss and the SGD Step Length," in *Proceedings of Learning Representations*, 2019.
[59] A. S. Golatkar, A. Achille, and S. Soatto, "Time Matters in Regularizing Deep Networks: Weight Decay and Data Augmentation Affect Early Learning Dynamics, Matter Little Near Convergence," in *Proceedings of Information Processing Systems*, 2019.
[60] G. Yan, H. Wang, X. Yuan, and J. Li, "DeFL: Defending Against Model Poisoning Attacks in Federated Learning via Critical Learning Periods Awareness," in *Proceedings of Artificial Intelligence and Innovative Applications of Artificial Intelligence and Educational Advances in Artificial Intelligence*, 2023.