Efficient and Light-Weight Federated Learning via Asynchronous Distributed Dropout

Chen Dun
Rice University

Mirian Hipolito
Microsoft Research

Chris Jermaine
Rice University

Dimitrios Dimitriadis
Microsoft Research

Anastasios Kyrillidis
Rice University

Abstract

Asynchronous learning protocols have regained attention lately, especially in the Federated Learning (FL) setup, where slower clients can severely impede the learning process. Herein, we propose `AsyncDrop`, a novel asynchronous FL framework that utilizes dropout regularization to handle device heterogeneity in distributed settings. Overall, `AsyncDrop` achieves better performance compared to state of the art asynchronous methodologies, while resulting in less communication and training time overheads. The key idea revolves around creating "submodels" out of the global model, and distributing their training to workers, based on device heterogeneity. We rigorously justify that such an approach can be theoretically characterized. We implement our approach and compare it against other asynchronous baselines, both by design and by adapting existing synchronous FL algorithms to asynchronous scenarios. Empirically, `AsyncDrop` reduces the communication cost and training time, while matching or improving the final test accuracy in diverse non-i.i.d. FL scenarios.

1 Introduction

Background on Federated Learning. Federated Learning (FL) [28, 24, 20] is a distributed learning protocol that has witnessed fast development the past demi-decade. FL deviates from the traditional distributed learning paradigms and allows the integration of edge devices —such as smartphones [38], drones [32], and IoT devices [29]— in the learning procedure. Yet, such real-life, edge devices are extremely *heterogeneous* [41]: they have drastically different specifications in terms of compute power, device memory and achievable communication bandwidths. Directly applying common *synchronized* FL algorithms –such as FedAvg and FedProx [24, 28] that require full model broadcasting and global synchronization– results often in a "stragglers" effect [30, 16, 39]; i.e., computationally powerful edge devices wait for slower ones during the synchronization step.

The ubiquitous synchronous training. One way to handle such issues is by utilizing *asynchrony* instead of *synchrony* in the learning process. To explain the main differences, let us first set up the background. In a synchronous distributed algorithm, a global model is usually stored at a central server and is broadcast periodically to all the participating devices. Then, each device performs local training steps on its own model copy, before the device sends the updated model to the central server. Finally, the central server updates the global model by aggregating the received model copies. This protocol is followed in most FL algorithms, including the well-established FedAvg [28], FedProx [24], FedNova [42] and SCAFFOLD [20]. The main criticism against synchronous learning could be that it often results in heavy communication/computation overheads and long idle/waiting times for workers.

Asynchrony and its challenges. The deployment of a asynchronous learning method is often convoluted. In the past decade, HogWild! [31, 27] has emerged as a general asynchronous distributed methodology, and has been applied initially in basic ML problems like sparse linear/logistic regression [51, 50, 15]. Ideally, based on sparsity arguments, each edge device can independently update parts of the global model –that overlap only slightly with the updates of other workers– in a lock-free fashion [31, 27]. This way, faster, more powerful edge workers do not suffer from idle waiting due to slower stragglers. Yet, the use of asynchrony has been a topic of dispute in distributed neural network training [7, 4]. Asynchronous training often suffers from lower accuracy as compared to synchronized SGD, which results in the dominance of synchronized SGD in neural network training [4].

Resurgence in asynchrony. Recently, asynchronous methods have regained popularity, mainly due to the interest in applying asynchronous motions within FL on edge devices: the heterogeneity of edge networks, the ephemeral nature of

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

the workers, the computational, communication and energy restriction of mobile devices are some impediments towards applying synchronous algorithms in realistic environments. Yet, traditional off-the-shelf asynchronous distributed algorithms still have issues, which might be exacerbated in the FL setting. As slower devices take longer local training time before updating the global model, this might result in inconsistent update schedules of the global model, compared to that of faster devices. This might have ramifications: i) For FL on i.i.d. data, this will cause the gradient staleness problem and result in convergence rate decrease; and, ii) on non-i.i.d. data, this will result in a significant drop in global model final accuracy.

As solutions, novel approaches on asynchronous FL propose weighted global aggregation techniques that take into consideration the heterogeneity of the devices [46, 5, 36]; yet, these methods often place a heavy computation/communication burden, as they rely on broadcasting full model updates to all the clients and/or the server. Other works monitor client speed to guide the training assignments [25, 3]. Finally, recent efforts propose semi-asynchronous methods, where participating devices are selected and buffered in order to complete a semi-synchronous global update periodically [16, 45]. A thorough discussion on the existing asynchronous methods in FL can be found in [47].

What is different in this work? As most algorithms stem from adapting asynchrony in synchronous FL, *one still needs to broadcast the full model to all devices, following a data parallel distributed protocol [11, 33], regardless of device heterogeneity*. This inspire us to ask a key question:

> *"Can we select submodels out of the global model and send these instead to each device, taking into account the device heterogeneity?"*

We answer this question affirmatively, by proposing a novel distributed dropout method for FL. We dub our method `AsyncDrop`. Our approach assigns different submodels to each device[1]; empirically, such a strategy decreases the required time to converge to an accuracy level, while preserving favorable final accuracy. This work attempts to reinstitute the discussion between synchrony and asynchrony in heterogeneous distributed scenarios, as in FL. Our idea is based on the ideas of HogWild! [31, 27] –in terms of sparse submodels– and Independent Subnetwork Training (IST) [49, 10, 26, 44] –where submodels are deliberately created for distribution, in order to decrease both computational and communication requirements.

Yet, we deviate from these works: i) The combination of HogWild! and IST ideas has not been stated and tested before this work, to the best of our knowledge. ii) While HogWild!-line of work provides optimization guarantees, we consider the non-trivial, non-convex neural network setting and provide theoretical guarantees for convergence; such a result combines tools from asynchronous optimization [31, 27], Neural Tangent Kernel assumptions [18], dropout theory analysis [26], and focuses on convolutional neural networks [23], deviating from fully-connected layer simplified scenarios. Finally, iii) we provide system-level algorithmic solutions for our approach, mirroring best-practices found during our experiments. Overall, the contributions of this work can be summarized as follows:

- We consider and propose *asynchronous distributed dropout* (`AsyncDrop`) for efficient large-scale FL. Our focus is on non-trivial, non-convex ML models –as in neural network training– and our framework provides specific engineering solutions for these cases in practice.
- We theoretically characterize and support our proposal with rigorous and non-trivial convergence rate guarantees. Currently, our theory assumes bounded delays; our future goal is to exploit recent developments that drop such assumptions [22]. Yet, our theory already considers the harder case of neural network training, which is often omitted in existing theory results.
- We provide specific implementation instances and share "best practices" for faster distributed FL in practice. As a side-product, our preliminary results include baseline asynchronous implementations of many synchronous methods (such as FedAvg, FedProx, and more), that are not existent currently, to the best of our knowledge.

2 Problem Setup and Challenges

Optimization in neural network training. We consider FL scenarios over *supervised* neural network training: i.e., we optimize a loss function $\ell(\cdot, \cdot)$ over a dataset, such that the model maps unseen data to their true labels, unless otherwise stated. For clarity, the loss $\ell(\mathbf{W}, \cdot)$ encodes both the loss metric and the neural architecture, with parameters $\mathbf{W}$. Formally, given a data distribution $\mathcal{D}$ and $\{\mathbf{x}_i, y_i\} \sim \mathcal{D}$, where $\mathbf{x}_i$ is a data sample, and y_i is its corresponding label, classical deep learning aims in finding $\mathbf{W}^\star$ as in:

$$\mathbf{W}^\star = \underset{\mathbf{W}\in\mathcal{H}}{\operatorname{argmin}} \left\{ \mathcal{L}(\mathbf{W}) := \frac{1}{n}\sum_{i=1}^{n} \ell\left(\mathbf{W}, \{\mathbf{x}_i, y_i\}\right) \right\},$$

where $\mathcal{H}$ denotes the model hypothesis class that "molds" the trainable parameters $\mathbf{W}$.

The minimization above can be achieved by using different approaches, but almost all training is accomplished via a variation of *stochastic gradient descent* (SGD) [35]. SGD modifies the current guess $\mathbf{W}_t$ using stochastic directions $\nabla\ell_{i_t}(\mathbf{W}_i) := \nabla\ell(\mathbf{W}_i, \{\mathbf{x}_{i_t}, y_{i_t}\})$. I.e., $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\nabla\ell_{i_t}(\mathbf{W}_t)$. Here, $\eta > 0$ is the learning rate, and i_t is a single or a mini-batch of examples. Most FL algorithms are based on these basic stochastic motions, like FedAvg [28], FedProx [24], FedNova [42] and SCAFFOLD [20].

[1] We consider both random assignment, as well as structured assignments, based on the computation power of the devices.

FL formulation. Let S be the total number of clients in a distributed FL scenario. Each client i has its own local data $\mathcal{D}_i$ such that the whole dataset satisfies $\mathcal{D} = \cup_i \mathcal{D}_i$, and usually $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset, \forall i \neq j$. The goal of FL is to find a global model $\mathbf{W}$ that achieves good accuracy on all data $\mathcal{D}$, by minimizing the following optimization problem:

$$\mathbf{W}^\star = \underset{\mathbf{W}\in\mathcal{H}}{\operatorname{argmin}} \left\{ \mathcal{L}(\mathbf{W}) := \frac{1}{S} \sum_{i=1}^{S} \ell\left(\mathbf{W}, \mathcal{D}_i\right) \right\},$$

where $\ell\left(\mathbf{W}, \mathcal{D}_i\right) = \frac{1}{|\mathcal{D}_i|} \sum_{\{\mathbf{x}_j, y_j\}\in\mathcal{D}_i} \ell\left(\mathbf{W}, \{\mathbf{x}_j, y_j\}\right)$. With a slight abuse of notation, $\ell\left(\mathbf{W}, \mathcal{D}_i\right)$ denotes the *local* loss function for user i, associated with a local model $\mathbf{W}_i$ (not indicated above), that gets aggregated with the models of other users. Herein, we consider both i.i.d. and non-i.i.d. cases, since local data distribution $\mathcal{D}_i$ can be heterogeneous and follow a non-i.i.d. distribution.

Algorithm 1 Meta Asynchronous FL

Parameters: T iters, S clients, l local iters., $\mathbf{W}$ as current global model, $\mathbf{W}_i$ as local model for i-th client, $\alpha \in (0, 1)$, η step size.

∞

$\mathbf{W} \leftarrow$ randomly initialized global model.
`//On each client i asynchronously:`
for $t = 0, \ldots, T-1$ **do**
  $\mathbf{W}_{i,t} \leftarrow \mathbf{W}$
  `//Train` $\mathbf{W}_i$ `for` l `iters. via SGD`
  for $j = 1, \ldots, l$ **do**
    $\mathbf{W}_{i,t} \leftarrow \mathbf{W}_{i,t} - \eta \frac{\partial \mathcal{L}}{\mathbf{W}_{i,t}}$
  end for
  `//Write local to global model`
  $\mathbf{W} \leftarrow (1-\alpha) \cdot \mathbf{W} + \alpha \cdot \mathbf{W}_{i,t}$
end for

Details of asynchronous training. An abstract description of how asynchronous FL operates is provided in Algorithm 1. In particular, given a number of server iterations T, each client i gets the updated global model $\mathbf{W}_t$ from the server, and further locally trains it using $\mathcal{D}_i$ for a number of local iterations l.[2] Asynchronous FL assumes each client has different computation power and communication bandwidth; this can be abstracted by different wall-clock times required to finish a number of local training iterations. Thus, when client i has completed its round, the updated model is shared with the server to be aggregated, before the next round of communication and computation starts for client i. *This is different from classical synchronous FL, where the global model is updated only when all participating clients finish (or time-out) certain local training iterations.*

[2]Details on the use of the optimizer, how it is tuned with respect to step size, mini-batch size, etc. are intentionally hidden at this point of the discussion.

3 Challenges in Asynchronous FL and Related Work

Challenges. Asynchronous steps often lead to inconsistent update schedules of the global model and are characterized by gradient staleness and drifting. Real-life FL applications include edge devices with limited communication and computation capabilities (e.g., how often and fast they can connect with the central server, and how powerful as devices they are). For instance, edge devices such as IoT devices or mobile phones [29], might only be able to communicate with the server within short time windows, due to network conditions or user permission policy.

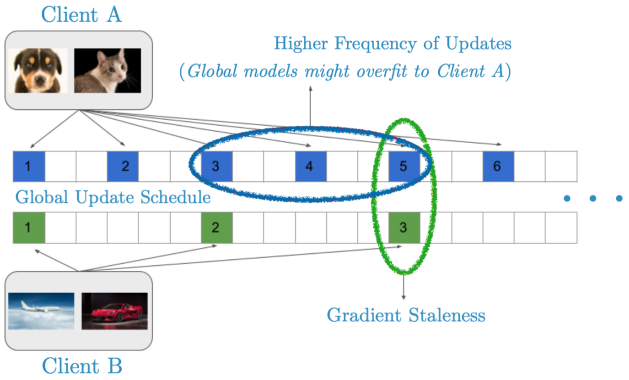

Figure 1: Potential issues in asynchronous FL.

Consider the toy setting in Figure 1. The two clients (Clients A and B) have a significantly different update schedule on the global model: Here, Client A has higher computational power or communication bandwidth –compared to client B– potentially leading to model drifting, lack of fair training and more severe gradient staleness. On top, consider these two clients having different local (non-i.i.d.) data distributions.

Related Work. The issue of model drifting due to data "non-iidness" is a central piece in FL research. Algorithms, such as FedProx [24], utilize regularization to constrain local parameters "drifting" away from previous global models.

The gradient staleness problem has been widely studied in asynchronous FL, like in [46, 5, 36, 25, 3]. These approaches can be summarized as weighted asynchronous FedAvg protocols, in which the weight of each local client update is proportional to the "capabilities" of the client. This should decrease the negative impact from stale gradients by slower clients. Semi-asynchronous methods have been proposed [16, 45]; yet, they require fast clients to wait until all other clients' updates are completed, in order to receive the updated model for the next round of local training.

Finally, numerous quantization [2, 48] and sparsification [1, 19] techniques have been proposed for reducing computation and communication costs in FL.

4 Asynchronous Distributed Dropout

(Distributed) Dropout. Dropout [40, 37, 12, 6] is a widely-accepted regularization technique in deep learning. The procedure of Dropout is as follows: per training round, a random mask over the parameters is generated; this mask is used to nullify part of the neurons in the neural network for this particular iteration. Variants of dropout include the drop-connect [40], multisample dropout [17], Gaussian dropout [43], and the variational dropout [21].

The idea of dropout has also been used in efficient distributed and/or FL scenarios. [13] introduces FjORD and the Ordered Dropout, a *synchronous* distributed dropout technique that leads to ordered, nested representation of knowledge in models, and enables the extraction of lower footprint submodels without the need of retraining. Such submodels are more suitable in client heterogeneity, as they adapt submodel's width to the client's capabilities. See also Nested Dropout [34] and HeteroFL [8].

Algorithm 2 Asynchronous dropout (`AsyncDrop`)

Parameters: T iters, S clients, l local iters., $\mathbf{W}$ as current global model, $\mathbf{W}_i$ as local model for i-th client, $\alpha \in (0,1)$, η step size.

∞

$\mathbf{W} \leftarrow$ randomly initialized global model.
`//On each client i asynchronously:`
for $t = 0, \dots, T-1$ **do**
  Generate mask $\mathbf{M}_{i,t}$
  $\mathbf{W}_{i,t} \leftarrow \mathbf{W}_t \odot \mathbf{M}_{i,t}$
  `//Train` $\mathbf{W}_{i,t}$ `for` l `iters. via SGD`
  for $j = 1, \dots, l$ **do**
    $\mathbf{W}_{i,t} \leftarrow \mathbf{W}_{i,t} - \eta \frac{\partial \mathcal{L}}{\mathbf{W}_{i,t}}$
  end for
  `//Write local to global model`
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t \odot (\mathbf{M}_{i,t})^c$
    $+((1-\alpha)\cdot \mathbf{W}_t + \alpha \cdot \mathbf{W}_{i,t}) \odot \mathbf{M}_{i,t}$
end for

Our proposal and main hypothesis. We focus on the *asynchronous version of distributed dropout.* We study theoretically whether asynchrony provably works in non-trivial non-convex scenarios *–as in training neural networks–* with random masks that generate submodels for each worker. The algorithm is described in Algorithm 2, dubbed as `AsyncDrop`, and is based on recent distributed protocols [49, 10, 26, 44]; key features are highlighted in teal-colored text. The main difference from Algorithm 1 is that Algorithm 2 splits the model vertically per iteration, where each submodel contains all layers of the neural network, but only with a (non-overlapping) subset of neurons being active in each layer. Multiple local SGD steps can be performed without the need for the workers to communicate. See also Figure 2 for a schematic representation of asynchronous distributed dropout for training a CNN.

Theoretical Results. We are interested in understanding whether such a combination of asynchronous computing and dropout techniques lead to convergence and favorable results: *given the variance introduced by both asynchronous updates and training of submodels, it is not obvious whether –and under which conditions– such a protocol would work.*

For ease of presentation and clarity of results, we analyse a one-hidden-layer CNN, and show convergence with random filter dropout. Consider a training dataset $(\mathbf{X}, \mathbf{y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where each $\mathbf{x}_i \in \mathbb{R}^{\hat{d} \times p}$ is an image and y_i being its label. Here, $\hat{d}$ is the number of input channels and p the number of pixels. Let q denote the size of the filter, and let m be the number of filters in the first layer. Based on previous work [9], we let $\hat{\phi}(\cdot)$ denote the patching operator with $\hat{\phi}(x) \in \mathbb{R}^{q\hat{d} \times p}$. Consider the first layer weight $\mathbf{W} \in \mathbb{R}^{m \times q\hat{d}}$, and second layer (aggregation) weight $\mathbf{a} \in \mathbb{R}^{m \times p}$. We assume that only the first layer weights $\mathbf{W}$ is trainable. The CNN trained on the means squared error has the form:

$$f(\mathbf{x}, \mathbf{W}) = \left\langle \mathbf{a}, \sigma\left(\mathbf{W}\hat{\phi}(\mathbf{x})\right), ; \right\rangle, \mathcal{L}(\mathbf{W}) = \|f(\mathbf{X}, \mathbf{W}) - \mathbf{y}\|_2^2,$$

where $f(\mathbf{x}, \cdot)$ denotes the output of the one-layer CNN for input $\mathbf{x}$, and $\mathcal{L}(\cdot)$ is the loss function. We use the ℓ_2-norm loss for simplicity. We make the following assumption on the training data and the CNN weight initialization.

Assumption 4.1 *(Training Data) Assume that for all $i \in [n]$, we have $\|\mathbf{x}_i\|_F = q^{-\frac{1}{2}}$ and $|y_i| \leq C$ for some constant C. Moreover, for all $i, i' \in [n]$ we have $\mathbf{x}_i \neq \mathbf{x}_{i'}$.*

Note that this can be satisfied by normalizing the data. For simplicity of the analysis, let $d := q\hat{d}$.

Assumption 4.2 *(Initialization) $\mathbf{w}_{0,i} \sim \mathcal{N}\left(0, \kappa^2 \mathbf{I}\right)$ and $a_{i,i'} \sim \left\{\pm \frac{1}{p\sqrt{m}}\right\}$ for $i \in [m]$ and $i' \in [p]$.*

In an asynchronous scenario, the neural network weight is updated with stale gradients due to the asynchronous updates, where δ_t is the delay at training step t. *We assume δ_t is bounded by a constant E.* Then, a simple version of gradient descent under these assumptions looks like:

$$\mathbf{W}_t = \mathbf{W}_t - \eta \nabla_{\mathbf{W}} \mathcal{L}\left(\mathbf{W}_{t-\delta_t}\right), \quad \delta_t \leq E,$$

where $\mathbf{W}_{t-\delta_t}$ indicates that the gradient is evaluated on a earlier version of the model parameters. Given the above, we provide the following guarantees:

Theorem 4.1 *Let $f(\cdot, \cdot)$ be a one-hidden-layer CNN with the second layer weight fixed. Let $\mathbf{u}_t$ abstractly represent the output of the model after t iterations, over the random selection of the masks. Let E denotes the maximum gradient delay/staleness. Let ξ denote the dropout rate ($\xi = 1$ dictates that all neurons are selected), and denote $\theta = 1 - (1-\xi)^S$*

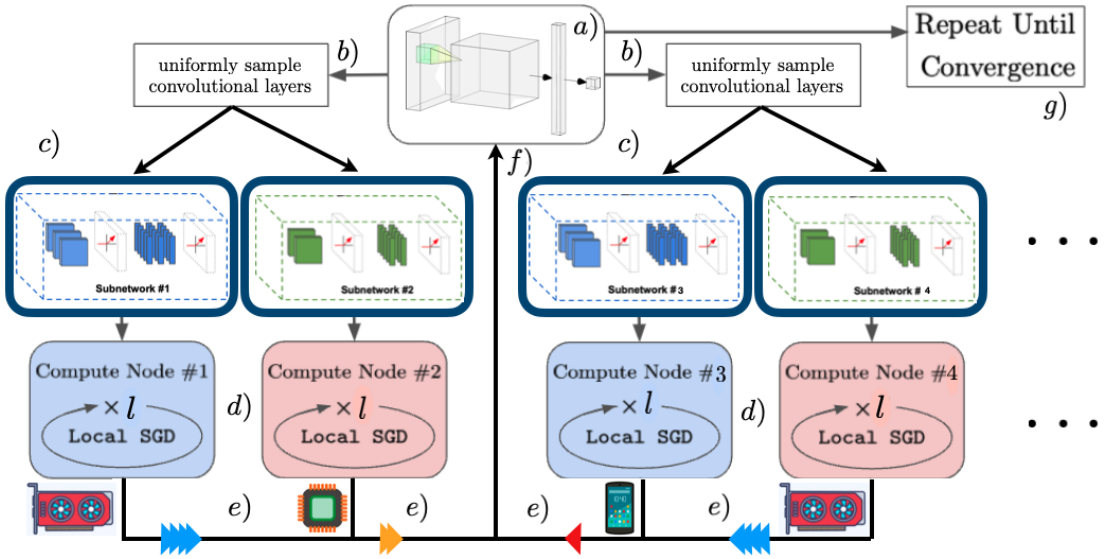

Figure 2: Schematic representation of `AsyncDropout`. $a)$ This is a simple representation of a CNN model. Our algorithm applies for arbitrary depth of CNNs (ResNets) as well as other architectures (MLPs, LSTMs, etc); here we restrict to a shallow CNN for illustration purposes. $b)$ Per request, random sub-sampled CNN models are created that result into different *subnetworks*. $c)$ These submodels are distributed to devices with different computational capabilities (here GPU, CPU, or a smartphone). $d)$ Without loss of generality, we assume that all devices train locally the submodel for l iterations. $e)$ However, each device finishes local training in different timestamps (shown as different colored arrows: red: slow speed; orange: moderate speed; blue: fast speed). $f)$ Yet, the global model is asynchronously updated and new submodels are created without global synchronization. $g)$ The above procedure is repeated till convergence.

the probability that a neuron is active in at least one subnetwork. Assume the number of hidden neurons satisfies $m = \Omega\left(\max\{\frac{n^4K^2}{\lambda_0^4\delta^2}\max\{n,d\},\frac{n}{\lambda_0}\}\right)$ *and the step size satisfies* $\eta = O\left(\frac{\lambda_0}{n^2}\right)$. *Let* κ *be a proper initialization scaling factor, and it is considered constant. We use* λ_0 *to denote the smallest eigenvalue of the Neural Tangent Kernel matrix. Let Assumptions 1 and 2 be satisfied. Then, the following convergence rate guarantee is proved to be satisfied:*

$$\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{u}_{t+1}-\mathbf{y}\|_2^2\right] \leq \left(1-\frac{\theta\eta\lambda_0}{4}\right)^t \|\mathbf{u}_0-\mathbf{y}\|_2^2 + O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2E^2}{n^2} + \frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4E^2}{m^4} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2 d}{m^3\lambda_0} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2\lambda_0E^2}{m^3} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2 d}{m^2\lambda_0} + \frac{n\kappa^2\left(\theta-\xi^2\right)}{S}\right)$$

Remark #1. This theorem states that the sum of the expected weight differences in the t-th iteration (i.e., $\mathbb{E}_{\mathbf{M}_t}[\|\mathbf{u}_{t+1}-\mathbf{y}\|_2^2]$) converges linearly to zero, as dictated by the red term $-\left(1-\frac{\theta\eta\lambda_0}{4}\right)^t\|\mathbf{u}_0-\mathbf{y}\|_2^2$– up to an error neighborhood, denoted with the Big-Oh notation term on the right hand side of the expression. Focusing on the latter, there are two types of additive errors: $i)$ the orange-colored terms origin from the dropout analysis: the term $1-\xi$ is often called as "dropout rate" (when $\xi=0$, no neurons are selected and the loss hardly decreases, while when $\xi=1$, all neurons are selected, and the orange-colored terms disappear). $ii)$ the violet-colored terms origin from the asynchronous analysis: when $E=0$ (i.e., we boil down to synchronous computations), these terms also disappear).

Remark #2. Beyond the above extreme cases, we observe that the error region terms can be controlled by algorithmic and model-design choices: e.g., when the size of the dataset n increases, the first term $\frac{\theta\eta\lambda_0^3\xi^2\kappa^2E^2}{n^2}$ can be controlled; for sufficiently wide neural network, the terms with m in the denominator can be made arbitrarily small; finally, notice that increasing the number of subnetworks S will drive the last term in the bound zero.

Vanilla asynchronous distributed dropout in practice. We test vanilla `AsyncDrop` with 25% Dropout rate in a

Table 1: Test accuracy of asynchronous FL baselines vs. AsyncDrop on non-i.i.d. CIFAR100 data over > 100 clients. We also report the time and communication overhead to reach a certain target accuracy: "Time for XX% Accuracy" denotes the second lowest test accuracy among all baselines as the target accuracy.

	Max. Test Accuracy	Time for 32% Accuracy	Time Overhead	Comm. Overhead
Sync. FedAvg	60.47	3890.6s	+69.38%	+42.2%
Async. FedAvg	32.47 ± 1.89	3062.5s	+33.00%	+15.56%
Async. Fed-Weighted-Avg	32.98 ± 1.71	3062.5s	+33.00%	+15.56%
Async. FedProx	35.75 ± 0.61	3062.5s	+33.33%	+15.56%
Async. FjORD	12.07 ± 0.83	N/A	N/A	N/A
AsyncDrop	**35.93** ± 0.92	**2296.8s**	**[Best]**	**[Best]**

FL setting with 104 heterogeneous clients and based on non-i.i.d. CIFAR100 dataset distribution. Beyond the extensions of FL baselines to the asynchronous setting (FedAvg, Fed-Weighted-Avg and FedProx), we further extend the work in [13] into Async. FjORD; further details in Sec 6. As shown in Table 1, AsyncDrop indeed shows improvements in three components: *final global model test accuracy, training time and communication cost*. These preliminary results demonstrate that AsyncDrop is on track to improve upon global model drifting, gradient staleness and computation/communication cost; all within the same single method.

5 Smart Partition in AsyncDrop for FL

Going beyond AsyncDrop. Despite theoretical support, AsyncDrop does not count device heterogeneity when submodels are assigned to the various workers. This could result into slower convergence rates and/or lower final model accuracy compared to other asynchronous methods [46, 5, 36, 25, 3] that carefully handle such cases. With non-i.i.d. data, more frequent updates by faster devices could lead to model drifting on local data. *These facts suggest a more careful handling of model splitting and model distribution among heterogeneous workers.*

Hetero AsyncDrop: An improved asynchronous solution. We propose to "balance" the contribution from different devices with the Hetero AsyncDrop method. Briefly, Hetero AsyncDrop assigns different weights to different devices, based on the rate update of the weights, and the computation power of the devices. The description of Hetero AsyncDrop is provided in Algorithm 3. We assign model weights that are less updated to the faster devices. Similarly, we assign model weights that are updated more often – and, thus, converging faster– to the slower devices. The premise behind such a protocol is that *all weights, eventually, will be updated/will converge with a similar rate.*

Hetero AsyncDrop: its ingredients. The above are encapsulated with a weight score function $v(\cdot)$, a device-capacity score function $\psi(\cdot)$ and the mask generator function $\varphi(\cdot)$ in Algorithm 3. The function $v(\cdot)$ quantifies the update speed of each weight. One simple score can be the norm of weight change: Given predefined grouping of weights

Algorithm 3 Hetero AsyncDrop for Asynchronous FL

Parameters: T iters, S clients, l local iters., $\mathbf{W}$ as current global model, $\mathbf{W}_i$ as local model for i-th client, η_g as global LR, $v(\cdot)$ weight score function, $\psi(i)$ computes the computation capacity of i-th worker, $\varphi(\mathbf{W}, \psi(i), v(\cdot))$ is the Smart Dropout function that creates the mask, based on worker capacity ψ and score v, $\alpha \in (0, 1)$.

$\mathbf{W} \leftarrow$ randomly initialized global model.
//On each client i asynchronously:
for $t = 0, \ldots, T-1$ **do**
  //For i-th fastest worker, $\varphi(\cdot)$ drops weights with i-th largest $v(\cdot)$ score
  Generate mask $\mathbf{M}_{i,t} = \varphi(\mathbf{W}_t, \psi(i), v(\cdot))$
  $\mathbf{W}_{i,t} \leftarrow \mathbf{W}_t \odot \mathbf{M}_{i,t}$
  //Train $\mathbf{W}_{i,t}$ for l iters. via SGD
  for $j = 1, \ldots, l$ **do**
    $\mathbf{W}_{i,t} \leftarrow \mathbf{W}_{i,t} - \eta_g \frac{\partial \mathcal{L}}{\mathbf{W}_{i,t}}$
  end for
  if i-th client is fastest **then**
    Update η_g
  end if
  //Write local to global model
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t \odot (\mathbf{M}_{i,t})^c + ((1-\alpha) \cdot \mathbf{W}_t + \alpha \cdot \mathbf{W}_{i,t}) \odot \mathbf{M}_{i,t}$
  //Update score q
  $v(\mathbf{W}_{t+1}^j) = \left\| \mathbf{W}_{t+1}^j - \mathbf{W}_0^j \right\|_1, \forall\, j \in \mathcal{J}$
end for

in set $\mathcal{J}$, define $\mathbf{W}_t^j$ for $j \in \mathcal{J}$ as the j-th weight/group of weights of global network at the t-th update. Then, the score function is selected as: $v(\mathbf{W}_t^j) = \left\| \mathbf{W}_t^j - \mathbf{W}_0^j \right\|_1$, i.e., the score function measures *how far from the initial values the j-th group of weights has moved.* In our experiments, we have tested grouping of weights $\mathcal{J}$ by filters and by layers. Further, we have some measure of the computation capacity of each device $\psi(i)$, and we order the devices in a descending order with respect to capabilities. The mask generator function $\varphi(\mathbf{W}, \psi(i), v(\cdot))$ generates masks that nullify weights in $\mathbf{W}$, based on the score function v over the

Table 2: Test accuracy of asynchronous FL baselines vs. (Hetero) AsyncDrop using a ResNet architecture on non-i.i.d CIFAR10, CIFAR100 and FMNIST data over > 100 clients. We report the time and communication overhead to reach a certain target accuracy: "Time for XX% Accuracy" denotes the second lowest test accuracy among all baselines as the target accuracy. Teal colored text indicates favorable performance; red colored text indicates high variance in performance.

CIFAR10	Max. Test Accuracy	Time for 35% Accuracy	Time Overhead	Comm. Overhead
Async. FedAvg	45.79 $\pm$ **7.9**	2105.5s	+33.34%	+15.56%
Async. Fed-Weighted-Avg	46.51 $\pm$ **6.8**	2105.5s	+33.34%	+15.56%
Async. FedProx	43.97 $\pm$ 1.35	2296.9s	+45.46%	+26.06%
Async. FjORD	23.14 $\pm$ 0.90	N/A	N/A	N/A
FedBuff	35.81 $\pm$ **11.83**	3012.9s	+89.98%	+35.75%
Hetero AsyncDrop	**50.67** $\pm$ 1.75	**1579.1s**	**[Best]**	**[Best]**
AsyncDrop	48.98 $\pm$ **3.87**	2009.7s	+27.27 %	+27.27%

CIFAR100	Max. Test Accuracy	Time for 32% Accuracy	Time Overhead	Comm. Overhead
Async. FedAvg	32.47 $\pm$ 1.89	3062.5s	+33.38%	+15.56%
Async. Fed-Weighted-Avg	32.98 $\pm$ 1.71	3062.5s	+33.38%	+15.56%
Async. FedProx	35.75 $\pm$ 0.61	3062.5s	+33.38%	+15.56%
Async. FjORD	12.07 $\pm$ 0.83	N/A	N/A	N/A
FedBuff	**41.91** $\pm$ **3.80**	4250.1s	+85.03%	+42.22%
Hetero AsyncDrop	37.26 $\pm$ 0.93	**2296.8s**	**[Best]**	**[Best]**
AsyncDrop	35.93 $\pm$ 0.93	**2296.8s**	**[Best]**	**[Best]**

FMNIST	Max. Test Accuracy	Time for 57% Accuracy	Time Overhead	Comm. Overhead
Async. FedAvg	62.47 $\pm$ **8.20**	787.5s	+33.33%	+25.00%
Async. Fed-Weighted-Avg	59.30 $\pm$ **10.44**	1181.3s	+100%	+87.50%
Async. FedProx	59.91 $\pm$ **7.32**	1050.0s	+77.78%	+66.67%
Async FjORD	21.98 $\pm$ **9.55**	N/A	N/A	N/A
FedBuff	57.03 $\pm$ **11.37**	1845.0s	+212.38%	+150.20%
Hetero AsyncDrop	**66.89** $\pm$ **5.36**	**590.6s**	**[Best]**	**[Best]**
AsyncDrop	60.02 $\pm$ **10.38**	787.5s	+33.33%	+33.33%

weights, and the device capacity list in $\psi(\cdot)$. The Hetero AsyncDrop strategy is that for the i-th fastest worker, we drop weights with i-th largest q score.

6 Experiments

Setup. We generate simulated FL scenarios with 104 clients/devices of diverse computation and communication capabilities. We implement clients as independent processes, each distributed on different GPUs with access to the same RAM space. We follow HogWild!'s distributed model [31]: i) we use a shared-memory system to store the global model; ii) each simulated client can update/read the global model in a fully lock free mode; and iii) each client transfers the local model to the assigned GPU for local training. We activate 8 clients at any given moment.

We use 25% dropout rate in (Hetero) AsyncDrop for CNN and MLP, while we use 12.5% for LSTM. *Even for such low dropout rates, the gains in training are obvious and significant, as we show in the experiments.* In the appendix, we provide ablation studies on how the dropout rate affects the performance of AsyncDrop-family of algorithms. We simulate the communication and computation savings by inserting shorter time delay, based on which we estimate the training time and communication cost.

Simulation of heterogeneous computations. We abstract heterogeneous computation and communication capabilities by "forcing" different delays after each training iteration. The delay time is inverse proportional to the intended capacity. In our experiments, we simulated 8 levels of computation and communication capabilities, that are evenly distributed between the slowest client and fastest clients. The difference between the slowest and the fastest clients is selected to be $\sim 5\times$. We make sure that, at any given moment, clients with diverse capacity are active. Finally, all clients with similar computation power shall have similarly biased local data distribution.[3]

Problem cases. We experiment on diverse neural network architectures and diverse types of learning tasks, including ResNets on Computer Vision datasets (CIFAR10, CIFAR100, FMNIST), MLPs on FMNIST dataset, and LSTMs on sentimental analysis (IMDB).

Baseline methods. For comparison, the baselines we consider are: i) the asynchronous FedAvg is the direct adaptation of FedAvg with asynchronous motions; ii) the asynchronous FedAvg with weighted aggregation represents

[3]This setting is to avoid the fastest clients with similar computation power cover all the data, which will reduce the problem into a trivial synchronous federated learning problem.

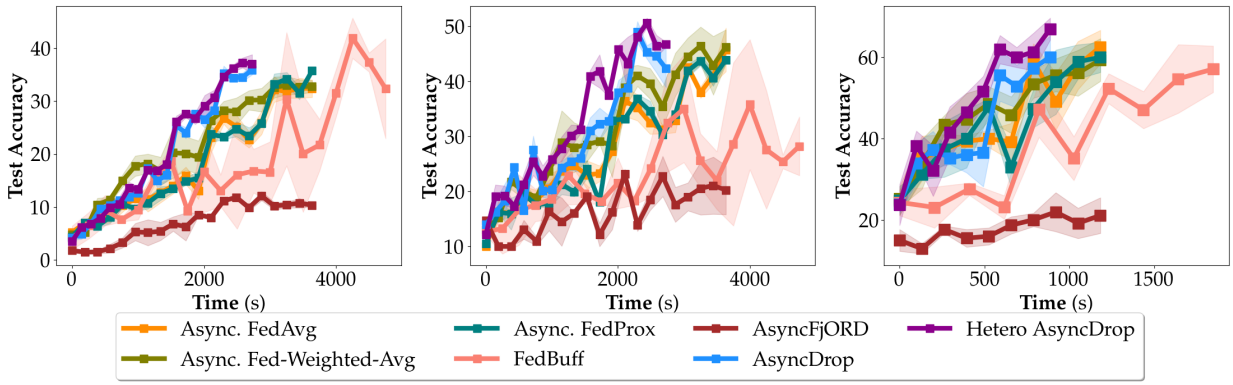

Figure 3: ResNet-based model. ***Left:*** CIFAR100 non-i.i.d.; ***Middle:*** CIFAR10, non-i.i.d.; ***Right:*** FMNIST, non-i.i.d.

the general approach of assigning devices different "importance", based on their capabilities [46, 5, 36]; iii) the asynchronous FjORD is our asynchronous adaptation of [14]; iv) the asynchronous FedProx adds an independent proximal loss to the local training loss of each device, in order to control gradient staleness and overfiting [24]; and v) FedBuff is a semi-asynchronous method which uses buffers for stale updates in a synchronized global scheme [30].

For all baselines and (Hetero) AsyncDrop on CIFAR10, CIFAR100, FMNIST, we set the local iterations at $l = 50$ while for IMDB, $l = 40$. For FedBuff, we set the buffer size to 4, which is half of the activated clients. We perform 3 trials with different random seeds. We report the maximum test accuracy, as well as the estimated time and communication cost to reach a certain target accuracy: *we select the second lowest test accuracy among all baselines as the target accuracy (third column in result tables).*

CNN-based results. We test (Hetero) AsyncDrop on ResNet34 and using CIFAR10, CIFAR100 and FMNIST datasets. For the CIFAR10 and CIFAR100 datasets, we train for 320 epochs, while for the FMNIST dataset, we train for 160 epochs. We stop the execution when the fastest client finishes all its epochs. As shown in Table 2, Hetero AsyncDrop shows non-trivial improvements in *final accuracy, training time and communication cost, simultaneously*. Hetero AsyncDrop shows lower accuracy compared with FedBuff in the CIFAR100 case; yet, it achieves up to 85% reduction in training time, due to the fact that FedBuff will require faster workers to wait until the buffer is filled to update the global model (this also justifies the up to $\sim 42.22\%$ reduction in total communication cost). Finally, we observe that (Hetero) AsyncDrop shows quite stable performance; in red color we indicate the variability of results over trials. The similar training time of some baselines to reach target accuracy is caused by epoch-wise testing, using same epoch-wise learning rate schedule and similar convergence rate as shown in Figure 3.

MLP-based and LSTM-based results. We adapt the (Hetero) AsyncDrop to the MLP model by applying the hidden neuron Dropout, which is similar to channel dropout in CNNs (160 epochs). For the LSTM and the IMDB sentimental analysis dataset (80 epochs), we create non-i.i.d. datasets based on different label distribution in each local training set. As shown in Table 1 in the Appendix, Hetero AsyncDrop achieves better performance overall over the MLP model, in terms of accuracy, training time and communication cost. As shown in Table 2 in the Appendix, for the LSTM model, Hetero AsyncDrop shows comparable accuracy with respect to other baselines, while achieving reduction in training time and communication cost in most cases. Our conjecture for the lower gain compared with other architectures is that LSTM-based (or even RNN-based) architectures might be difficult to our proposed dropout score mechanism, as the update of each network parameter is the average of several virtual parameters in the unrolled network.

7 Concluding Remarks

We present (Hetero) AsyncDrop, a novel algorithm for asynchronous distributed neural network training in the FL scenario. AsyncDrop operates by asynchronously creating submodels out of the global model, in order to train those independently on heterogeneous devices. These models are asynchronously aggregated into the global model. By only communicating and training submodels over edge workers, AsyncDrop reduces the communication and local training cost. We demonstrate the impact of AsyncDrop on MLPs, ResNets and LSTMs over non-i.i.d. distributions of CIFAR10, CIFAR100, FMNIST and IMDB datasets. Additional experiments in the appendix include ablation analysis on how dropout ratio, local iteration length and dropout strategy affect the final performance.

We aim to extend AsyncDrop to other network architectures, such as Transformers. Further, AsyncDrop is fully-compatible with various gradient compression methods, which could potentially further improve the performance. We will investigate the prospect of fully integrating such compression methods within AsyncDrop, both during training and communication phases in future work.

References

[1] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 440–445, 2017. [Cited on page 3.]

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Advances in Neural Information Processing Systems*, 30, 2017. [Cited on page 3.]

[3] Zheng Chai, Yujing Chen, Liang Zhao, Yue Cheng, and Huzefa Rangwala. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. *ArXivorg*, 2020. [Cited on pages 2, 3, and 6.]

[4] Jianmin Chen, Xinghao Pan, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016. [Cited on page 1.]

[5] Yang Chen, Xiaoyan Sun, and Yaochu Jin. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE transactions on neural networks and learning systems*, 31(10):4229–4238, 2019. [Cited on pages 2, 3, 6, and 8.]

[6] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015. [Cited on page 4.]

[7] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012. [Cited on page 1.]

[8] Enmao Diao, Jie Ding, and Vahid Tarokh. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020. [Cited on page 4.]

[9] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019. [Cited on page 4.]

[10] Chen Dun, Cameron R Wolfe, Christopher M Jermaine, and Anastasios Kyrillidis. ResIST: Layer-wise decomposition of ResNets for distributed training. In *Uncertainty in Artificial Intelligence*, pages 610–620. PMLR, 2022. [Cited on pages 2 and 4.]

[11] P. Farber and K. Asanovic. Parallel neural network training on multi-spert. In *Proceedings of 3rd International Conference on Algorithms and Architectures for Parallel Processing*, pages 659–666, 1997. [Cited on page 2.]

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016. [Cited on page 4.]

[13] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021. [Cited on pages 4 and 6.]

[14] Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos I. Venieris, and Nicholas D. Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout, 2021. [Cited on page 8.]

[15] Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. PaSSCODE: Parallel asynchronous stochastic dual coordinate descent. In *International Conference on Machine Learning*, pages 2370–2379. PMLR, 2015. [Cited on page 1.]

[16] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems*, 4:814–832, 2022. [Cited on pages 1, 2, and 3.]

[17] Hiroshi Inoue. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*, 2019. [Cited on page 4.]

[18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018. [Cited on page 2.]

[19] Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. *Advances in Neural Information Processing Systems*, 31, 2018. [Cited on page 3.]

[20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning, 2019. [Cited on pages 1 and 2.]

[21] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015. [Cited on page 4.]

[22] Anastasia Koloskova, Sebastian U Stich, and Martin Jaggi. Sharper convergence guarantees for asynchronous SGD for distributed and federated learning. *arXiv preprint arXiv:2206.08307*, 2022. [Cited on page 2.]

[23] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. [Cited on page 2.]

[24] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2018. [Cited on pages 1, 2, 3, and 8.]

[25] Xingyu Li, Zhe Qu, Bo Tang, and Zhuo Lu. Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients. *arXiv preprint arXiv:2102.06329*, 2021. [Cited on pages 2, 3, and 6.]

[26] Fangshuo Liao and Anastasios Kyrillidis. On the convergence of shallow neural network training with randomly masked neurons. *arXiv preprint arXiv:2112.02668*, 2021. [Cited on pages 2 and 4.]

[27] Ji Liu, Steve Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. In *International Conference on Machine Learning*, pages 469–477. PMLR, 2014. [Cited on pages 1 and 2.]

[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. [Cited on pages 1 and 2.]

[29] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021. [Cited on pages 1 and 3.]

[30] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pages 3581–3607. PMLR, 2022. [Cited on pages 1 and 8.]

[31] Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent, 2011. [Cited on pages 1, 2, and 7.]

[32] Yuben Qu, Haipeng Dai, Yan Zhuang, Jiafa Chen, Chao Dong, Fan Wu, and Song Guo. Decentralized federated learning for UAV networks: Architecture, challenges, and opportunities. *IEEE Network*, 35(6):156–162, 2021. [Cited on page 1.]

[33] R. Raina, A. Madhavan, and A. Ng. Large-scale deep unsupervised learning using graphics processors. In *ICML*, pages 873–880. ACM, 2009. [Cited on page 2.]

[34] Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pages 1746–1754. PMLR, 2014. [Cited on page 4.]

[35] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. [Cited on page 2.]

[36] Guomei Shi, Li Li, Jun Wang, Wenyan Chen, Kejiang Ye, and ChengZhong Xu. Hysync: Hybrid federated learning with effective synchronization. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 628–633, 2020. [Cited on pages 2, 3, 6, and 8.]

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. [Cited on page 4.]

[38] Branislav Stojkovic, Jonathan Woodbridge, Zhihan Fang, Jerry Cai, Andrey Petrov, Sathya Iyer, Daoyu Huang, Patrick Yau, Arvind Sastha Kumar, Hitesh Jawa, et al. Applied federated learning: Architectural design for robust and efficient learning in privacy aware settings. *arXiv preprint arXiv:2206.00807*, 2022. [Cited on page 1.]

[39] Rashish Tandon, Qi Lei, Alexandros G Dimakis, and Nikos Karampatziakis. Gradient coding: Avoiding stragglers in distributed learning. In *International Conference on Machine Learning*, pages 3368–3376. PMLR, 2017. [Cited on page 1.]

[40] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013. [Cited on page 4.]

[41] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021. [Cited on page 1.]

[42] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020. [Cited on pages 1 and 2.]

[43] Sida Wang and Christopher Manning. Fast dropout training. In *international conference on machine learning*, pages 118–126. PMLR, 2013. [Cited on page 4.]

[44] Cameron R Wolfe, Jingkang Yang, Arindam Chowdhury, Chen Dun, Artun Bayer, Santiago Segarra, and Anastasios Kyrillidis. GIST: Distributed training for large-scale graph convolutional networks. *arXiv preprint arXiv:2102.10424*, 2021. [Cited on pages 2 and 4.]

[45] Wentai Wu, Ligang He, Weiwei Lin, Rui Mao, Carsten Maple, and Stephen Jarvis. Safa: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5):655–668, 2020. [Cited on pages 2 and 3.]

[46] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization, 2019. [Cited on pages 2, 3, 6, and 8.]

[47] Chenhao Xu, Youyang Qu, Yong Xiang, and Longxiang Gao. Asynchronous federated learning on heterogeneous devices: A survey, 2021. [Cited on page 2.]

[48] Yue Yu, Jiaxiang Wu, and Longbo Huang. Double quantization for communication-efficient distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019. [Cited on page 3.]

[49] Binhang Yuan, Cameron R Wolfe, Chen Dun, Yuxin Tang, Anastasios Kyrillidis, and Chris Jermaine. Distributed learning of fully connected neural networks using independent subnet training. *Proceedings of the VLDB Endowment*, 15(8):1581–1590, 2022. [Cited on pages 2 and 4.]

[50] Hyokun Yun, Hsiang-Fu Yu, Cho-Jui Hsieh, SVN Vishwanathan, and Inderjit Dhillon. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *arXiv preprint arXiv:1312.0193*, 2013. [Cited on page 1.]

[51] Yong Zhuang, Wei-Sheng Chin, Yu-Chin Juan, and Chih-Jen Lin. A fast parallel SGD for matrix factorization in shared memory systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 249–256, 2013. [Cited on page 1.]

Supplementary material: Efficient and Light-Weight Federated Learning via Asynchronous Distributed Dropout

1 Experimental Results using `AsyncDrop` for the MLP and LSTM architectures

We adapt the (`Hetero`) `AsyncDrop` to the MLP and LSTM model by applying the hidden neuron Dropout, which is similar to channel dropout in CNNs. For the LSTM and the IMDB sentimental analysis dataset, we create non-i.i.d. datasets based on different label distribution in each local training set. For the FMNIST experiments, we train for 160 epochs, while for the IMDB experiments, we train for 80 epochs. As shown in Table 1, `Hetero AsyncDrop` achieves better performance overall over the MLP model, in terms of accuracy, training time and communication cost. As shown in Table 2, for the LSTM model, `Hetero AsyncDrop` shows comparable accuracy with respect to other baselines, while achieving reduction in training time and communication cost in most cases. Our conjecture for the lower gain compared with other architectures is that LSTM-based (or even RNN-based) architectures might be difficult to our proposed dropout score mechanism, as the update of each network parameter is the average of several virtual parameters in the unrolled network.

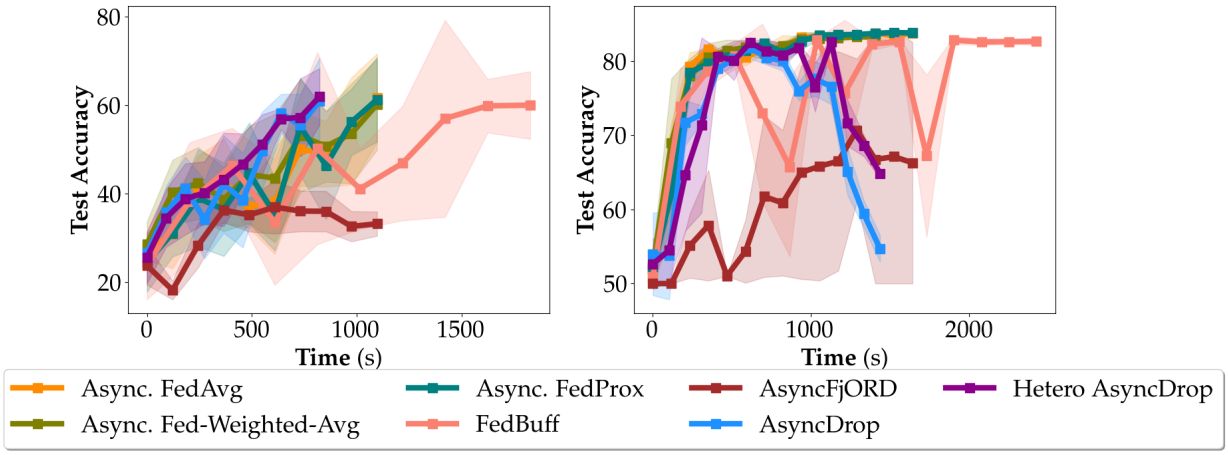

Figure 1: ***Left:*** MLP FMNIST non-i.i.d.; ***Right:*** LSTM IMDB, non-i.i.d.

Table 1: Test accuracy of asynchronous FL baselines vs. (`Hetero`) `AsyncDrop` using a MLP architecture on non-i.i.d MNIST data over > 100 clients. We report the time and communication overhead to reach a certain target accuracy: "Time for `XX`% Accuracy" denotes the second lowest test accuracy among all baselines as the target accuracy.

	Max. Test Accuracy	Time for 59.95% Accuracy	Time Overhead	Comm. Overhead
Async. FedAvg	$61.59 \pm$ **10.02**	1096.1s	+33.33%	+28.73%
Async. Fed-Weighted-Avg	$60.14 \pm$ **10.28**	1096.1s	+33.33%	+28.73%
Async. FedProx	$61.25 \pm$ **9.54**	1096.1s	+33.33%	+28.73%
Async. FjORD	36.92 ± 6.00	N/A	N/A	N/A
FedBuff	$59.96 \pm$ **7.64**	1827.3s	+166.50%	+83.90%
`Hetero AsyncDrop`	61.90 ± 6.39	**822.5s**	**[Best]**	**[Best]**
`AsyncDrop`	$60.85 \pm$ **9.78**	**822.5s**	**[Best]**	**[Best]**

Table 2: Test accuracy of asynchronous FL baselines vs. (Hetero) AsyncDrop using a LSTM architecture on non-i.i.d IMDB data over > 100 clients. We report the time and communication overhead to reach a certain target accuracy: "Time for XX% Accuracy" denotes the second lowest test accuracy among all baselines as the target accuracy.

	Max. Test Accuracy	Time for 82% Accuracy	Time Overhead	Comm. Overhead
Async. FedAvg	83.73 ± 0.18	1875.6s	+52.38%	+32.06%
Async. Fed-Weighted-Avg	83.26 ± 0.33	1640.6s	+33.33%	+15.56%
Async. FedProx	83.83 ± 0.20	1406.3	+14.28%	**-0.95%**
Async. FjORD	70.68 ± 9.62	N/A	N/A	N/A
FedBuff	82.81 ± 0.30	1038.1s	+68.71%	+14.28%
Hetero AsyncDrop	82.60 ± 0.57	**1230.4s**	**[Best]**	-
AsyncDrop	81.70 ± 1.45	N/A	N/A	N/A

2 Ablation Study

Dropout Ratio for Hetero AsyncDrop. In this section we analyze how dropout ratio affects the final accuracy and training time. As shown in Table 3, low dropout rates result in longer training time and lower final accuracy. Lower final accuracy is mainly due to the fact that AsyncDrop with dropout work as a regularization technique that works favorably towards better final accuracy. On the other hand, higher dropout rates cause relative reduction in training time, but significant reduction in the final accuracy. This shows dropping too many parameters causes the training of local model unstable, reducing the final accuracy of the global model.

Table 3: Test accuracy of Hetero AsyncDrop with different dropout rate on non-i.i.d. CIFAR100 datasets.

Dropout Ratio	0%	25%	50%
Test Accuracy	32.47 ± 1.89	37.26 ± 0.93	23.95 ± 1.07
Time for Accuracy to 23%	2105.4s	1579.1s	1531.2s

Local Iteration Length for Hetero AsyncDrop. In this section we analyze how the number of local iterations (l) affect the final accuracy and training time. Ideally, longer local training iteration will decrease the communication frequency and, thus, the communication and time cost. [5, 2] However, as shown in Figure 2, increasing local training iterations cause significant decrease in final accuracy, which might be due to i) increased gradient staleness from slower clients; ii) more severe model drifting. In future research, we will study how to improve Hetero AsyncDrop in order to be more robust for higher l values.

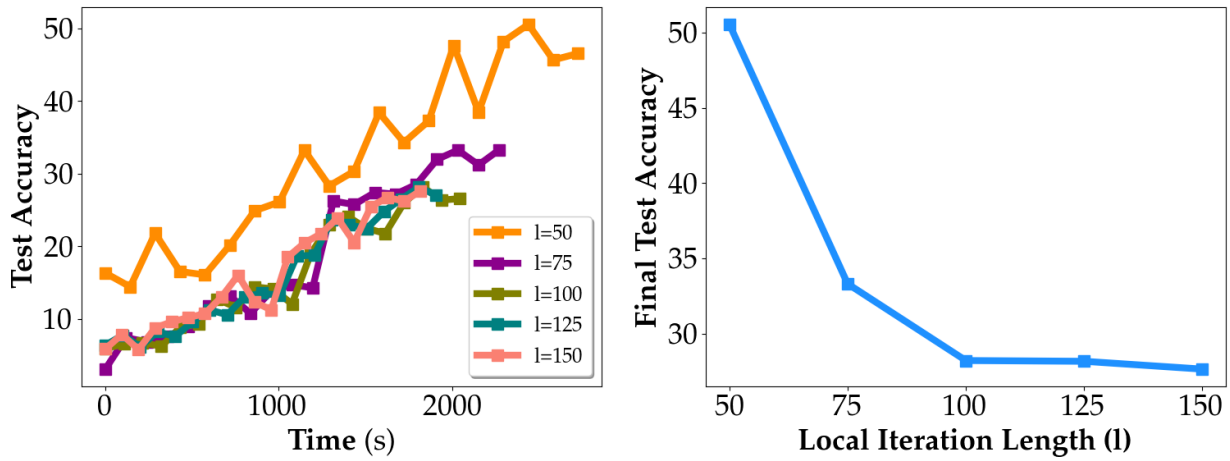

Figure 2: Test Accuracy of Hetero AsyncDrop with different local iteration length on CIFAR100, non i.i.d

The effectiveness of Smart Partition in Hetero AsyncDrop. As shown in Table 4, on all datasets and all network architectures, Hetero AsyncDrop provides improvements on the final accuracy, while decreasing the variance. This indicates that Hetero AsyncDrop could be a more stable method that balances heterogeneous clients and data. By

interpreting current experiments, `Hetero AsyncDrop` gives the most gain on ResNet (or CNN model family), which utilizes a channel dropout method.

On the other hand, in both MLP and LSTM, we dropout output hidden neurons and the network parameters accordingly. As filter channels in ResNet are considered to be more semantically disentangled from each other, compared to hidden neurons in fully connected layers, `Hetero AsyncDrop`'s score for filters in ResNet capture better which filters are more frequently/significantly updated (and thus biased towards the faster clients). Thus, our scoring function $q(\cdot)$ targets which parameters to drop in ResNet architectures. We leave to future research on how to improve our proposed mechanism on MLP and LSTM architectures, potentially by using a more structured dropout strategy.

Table 4: Test accuracy of `Hetero AsyncDrop` versus `AsyncDrop`

	`Hetero AsyncDrop`	`AsyncDrop`	Relative Accuracy Gain
ResNet+CIFAR10	50.67 ± 1.75	48.98 ± 3.87	+3.45%
ResNet+CIFAR100	37.26 ± 0.93	35.93 ± 1.07	+2.98%
ResNet+FMNIST	66.89 ± 5.36	60.02 ± 10.38	+11.45%
MLP+FMNIST	61.90 ± 6.39	60.85 ± 9.78	+1.72%
LSTM+IMDB	82.60 ± 0.57	81.70 ± 1.45	+1.10%

Layerwise Smart Partition for `AsyncDrop` on ResNet. For ResNet architectures, another commonly used dropout strategy is that of layerwise dropout [3, 2]. We adapt `Hetero AsynDrop` using layerwise Smart Parition and dropout. The main difference from original `Hetero AsyncDrop` algorithm is that we consider each layer as a single unit for dropout. The Smart Partition score function calculates the total norm change of all parameters in each layer and ranks layers accordingly. Finally, for the i-th fastest worker, we drop layers with the i-th largest q score. In experiments, we set the dropout rate to be 25% to keep the submodel size similar to the original `AsyncDrop` experimebts. As shown in Figure 3, Layerwise `Hetero AsyncDrop` results in a non-negligible decease in performance, with respect to the final accuracy. This might show layer dropping causes increased variance and instability, especially in asynchronous training.

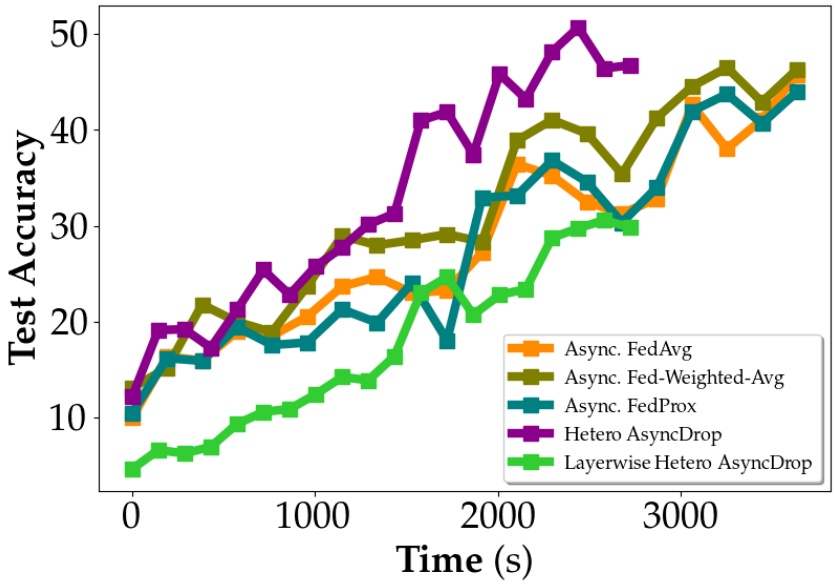

Figure 3: Test Accuracy of Layerwise `Hetero AsyncDrop` versus all baselines on CIFAR100 non- i.i.d.

The robustness of `AsyncDrop` to "full stragglers". We perform additional experiments to demonstrate `AsyncDrop` is very robust to "full stragglers". We use $>$100 clients and gradually (linear w.r.t. training time) permanently make random clients go offline until 8 clients left. Here, we evenly drop off clients to ensure the heterogeneity of device and data at any moment of training. As shown in Table 5 and Figure 4, `Hetero AsyncDrop` shows non-trivial improvements in final accuracy, training time and communication cost, simultaneously. Although `Hetero AsyncDrop` has slightly lower final accuracy compared with FedBuff, it achieves up to 74.14% reduction in total training time and has significantly lower variance.

r

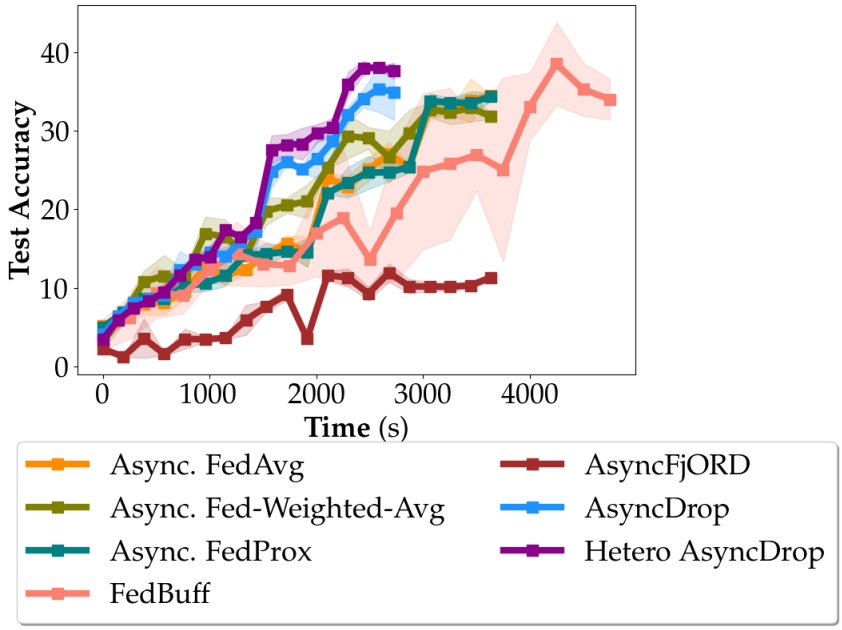

Figure 4: `AsyncDrop` on non-i.i.d CIFAR100 with random clients permanently going offline

Table 5: Test accuracy of asynchronous FL baselines vs. (`Hetero`) `AsyncDrop` using a ResNet architecture on non-i.i.d CIFAR100 data over > 100 clients with clients randomly permenantly going offline.

`CIFAR100`	Max. Test Accuracy	Time for 32.9% Accuracy	Time Overhead	Comm. Overhead
Async. FedAvg	34.52 ± 0.04	3062.5s	+33.38%	+15.56%
Async. Fed-Weighted-Avg	32.92 ± 1.75	3445.3s	+50.00%	+30.00%
Async. FedProx	34.34 ± 0.59	3062.5s	+33.33%	+15.56%
Async. FjORD	11.93 ± 1.15	N/A	N/A	N/A
FedBuff	**38.52** $\pm$ **5.28**	4000.1s	+74.14%	+24.44%
`Hetero AsyncDrop`	38.03 ± 0.18	**2290.1s**	**[Best]**	**[Best]**
`AsyncDrop`	35.33 ± 2.93	2440.4s	+6.25%	+6.25%

3 Detailed Mathematical Formulation of `AsyncDrop`

For a vector $\mathbf{v}$, $\|\mathbf{v}\|_2$ denotes its Euclidean (ℓ_2) norm. For a matrix $\mathbf{V}$, $\|\mathbf{V}\|_F$ denotes its Frobenius norm. We use $\mathcal{P}(\cdot)$ to denote the probability of an event, and $\mathbb{I}\{\cdot\}$ to denote the indicator function. For two vectors $\mathbf{v}_1, \mathbf{v}_2$, we use the simplified notation $\mathbb{I}\{\mathbf{v}_1; \mathbf{v}_2\} := \mathbb{I}\{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle \geq 0\}$. Given that the mask in iteration t is $\mathbf{M}_t$, we denote $\mathbb{E}_{[\mathbf{M}_t]}[\cdot] = \mathbb{E}_{\mathbf{M}_0,\ldots,\mathbf{M}_t}[\cdot]$.

Recall that the CNN considered in this paper has the form:

$$f(\mathbf{x}, \theta) = \left\langle \mathbf{a}, \sigma\left(\mathbf{W}_1 \hat{\phi}(\mathbf{x})\right) \right\rangle.$$

Denote $\hat{\mathbf{x}} = \hat{\phi}(\mathbf{x})$. Essentially, this patching operator applies to each channel, with the effect of extending each pixel to a set of pixels around it. So we denote $\hat{\mathbf{x}}_i^{(j)} \in \mathbb{R}^{q\hat{d}}$ as the extended jth pixel across all channels in the ith sample. For each transformed sample, we have that $\|\hat{\mathbf{x}}_i\|_F \leq \sqrt{q}\|\mathbf{x}_i\|_F$. We simplify the CNN output as¨

$$f(\hat{\mathbf{x}}, \mathbf{W}) = \langle \mathbf{a}, \sigma(\mathbf{W} \otimes \hat{\mathbf{x}}) \rangle = \sum_{r=1}^{m_1} \sum_{j=1}^{p} a_{rj} \sigma\left(\left\langle \hat{\mathbf{x}}^{(j)}, \mathbf{w}_r \right\rangle\right).$$

In this way, the formulation of CNN reduces to MLP despite a different form of input data $\hat{\mathbf{x}}$ and an additional dimension of aggregation in the second layer. We consider training the neural network f on the mean squared error (MSE):

$$\mathcal{L}(\mathbf{W}) = \left\| f\left(\hat{\mathbf{X}}, \mathbf{W}\right) - \mathbf{y} \right\|_2^2 = \sum_{i=1}^{n} \left(f(\hat{\mathbf{x}}_i, \mathbf{W}) - y_i\right)^2.$$

Similar to HogWild! training is fully lock free and thus cannot prevent several clients from updating the global model at same time, in order to be closer to real-world case, in theoretical analysis, we extend `AsyncDrop` by allowing $S \geq 1$ clients –with same computation speed and thus same global model update schedule– to update the global model simultaneously by averaging their updates. This can be reduced back to original `AsyncDrop`, by simply setting $S = 1$ without changing any part of the analysis.

We assume that we can group all clients by their computation/communication power, such that, in each group, there are exactly $S \geq 1$ clients with the same power; thus, with the same global model update schedule. Each worker will be independently assigned to different dropout masks to be applied on the global model. All updates from local clients in the same group will be averaged.

In math, the subnetwork on client s by filter-wise partition is given by:

$$f_{\mathbf{m}^{(s)}}\left(\hat{\mathbf{x}}, \mathbf{W}\right) = \sum_{r=1}^{m_1} \sum_{j=1}^{p} m_r^{(s)} a_{rj} \sigma\left(\left\langle \hat{\mathbf{x}}^{(j)}, \mathbf{w}_r \right\rangle\right).$$

Trained on the regression loss, the surrogate gradient on current network parameters is given by:

$$\nabla_{\mathbf{w}_r} \mathcal{L}_{\mathbf{m}^{(s)}}\left(\mathbf{W}\right) = m_r^{(s)} \sum_{i=1}^{n} \sum_{j=1}^{p} \left(f_{\mathbf{m}^{(s)}}\left(\hat{\mathbf{x}}_i, \mathbf{W}\right) - y_i\right) a_{rj} \hat{\mathbf{x}}_i^{(j)} \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_r\right\}.$$

We correspondingly scale the whole network function:

$$f\left(\hat{\mathbf{x}}, \mathbf{W}\right) = \xi \sum_{r=1}^{m} \sum_{j=1}^{p} a_{rj} \sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_r \right\rangle\right).$$

Assuming it is training on the MSE, we write out its gradient as:

$$\nabla_{\mathbf{w}_r} \mathcal{L}\left(\mathbf{W}\right) = \xi \sum_{i=1}^{n} \sum_{j=1}^{n} \left(f\left(\hat{\mathbf{x}}_i, \mathbf{W}\right) - y_i\right) a_{rj} \hat{\mathbf{x}}_i^{(j)} \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_r\right\}.$$

In this work, we consider the ASYNDROP with one step of local iteration training and $\delta_t \leq E$ gradient delay caused by asynchronous training, given by:

$$\mathbf{w}_{r,t+1} = \mathbf{w}_{r,t} - \eta \frac{N_{r,t-\delta_t}^{\perp}}{N_{r,t-\delta_t}} \sum_{s=1}^{S} \nabla_{\mathbf{w}_r} \mathcal{L}_{\mathbf{m}_{t-\delta_t}^{(s)}}\left(\mathbf{W}_{r,t-\delta_t}\right).$$

Here, let $N_{r,t} = \max\left\{\sum_{s=1}^{S} m_{r,t}^{(s)}, 1\right\}$, and $N_{r,t}^{\perp} = \min\left\{\sum_{s=1}^{S} m_{r,t}^{(s)}, 1\right\}$. Intuitively, $N_{r,t}$ denote the "normalizer" that we will divide the sum of the gradients from all subnetworks with, and $N_{r,t}^{\perp}$ denote the indicator of whether filter r is trained in at least one subnetwork. Let $\theta = \mathcal{P}\left(N_{r,t}^{\perp} = 1\right) = 1 - (1-\xi)^p$, denoting the probability that at least one of $\left\{m_{r,t}^{(s)}\right\}_{s=1}^{S}$ is one. Denote $u_t^{(i)} = f\left(\hat{\mathbf{x}}_i, \mathbf{W}_t\right)$. For further convenience of our analysis, we define

$$\tilde{u}_{r,t}^{(i)} = \frac{N_{r,t}^{\perp}}{N_{r,t}} \sum_{s=1}^{S} m_{r,t}^{(s)} \hat{u}_t^{(s,i)}; \quad \mathbf{g}_{r,t} = \frac{N_{r,t}^{\perp}}{N_{r,t}} \sum_{s=1}^{S} \nabla_{\mathbf{w}_r} \mathcal{L}_{\mathbf{m}_t^{(s)}}\left(\mathbf{W}_t\right).$$

Then the `AsyncDrop` training has the form

$$\mathbf{w}_{r,t+1} = \mathbf{w}_{r,t} - \eta \mathbf{g}_{r,t-\delta_t}; \quad \mathbf{g}_{r,t} = \sum_{i=1}^{n} \sum_{j=1}^{p} a_{rj} \left(\tilde{u}_{r,t}^{(i)} - N_{r,t}^{\perp} y_i\right) \hat{\mathbf{x}}_i^{(j)} \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}.$$

Suppose that Assumptions 1 and 2 in main text hold. Then for all $i \in [n]$ we have $\left\|\mathbf{x}_i\right\|_F = q^{\frac{1}{2}}$, and for all $i, i' \in [n]$ such that $i \neq i'$, we have $\mathbf{x}_i \neq \mathbf{x}_{i'}$. As in previous work [1], we have $\left\|\hat{\mathbf{x}}_i\right\|_F =\leq \sqrt{q} \left\|\mathbf{x}_i\right\|_F \leq 1$. Thus, for all $j \in [p]$ we have $\left\|\hat{\mathbf{x}}_i^{(j)}\right\| \leq 1$. Moreover, since $\mathbf{x}_i \neq \mathbf{x}_{i'}$ for $i \neq i'$, we then have $\hat{\mathbf{x}}_i \neq \hat{\mathbf{x}}_{i'}$, which implies that $\hat{\mathbf{x}}_i^{(j)} \neq \hat{\mathbf{x}}_{i'}^{(j)}$ for all $i \neq i'$ and $j \in [p]$.

4 Proof of `AsyncDrop` Convergence

In this section, our goal is to prove the convergence of extended `AsyncDrop`. We first state the extended version here, and proceed proving it.

Theorem 4.1 *Let $f(\cdot,\cdot)$ be a one-hidden-layer CNN with the second layer weight fixed. Let $\mathbf{u}_t$ abstractly represent the output of the model after t iterations, over the random selection of the masks. Let E denotes the maximum gradient delay/staleness. Let ξ denote the dropout rate ($\xi = 1$ dictates that all neurons are selected), and denote $\theta = 1-(1-\xi)^S$ the probability that a neuron is active in at least one subnetwork. Assume the number of hidden neurons satisfies $m = \Omega\left(\max\{\frac{n^4K^2}{\lambda_0^4\delta^2}\max\{n,d\}, \frac{n}{\lambda_0}\}\right)$ and the step size satisfies $\eta = O\left(\frac{\lambda_0}{n^2}\right)$. Let κ be a proper initialization scaling factor, and it is considered constant. We use λ_0 to denote the smallest eigenvalue of the Neural Tangent Kernel matrix. Let Assumptions 1 and 2 be satisfied. Then, the following convergence rate guarantee is proved to be satisfied:*

$$\begin{aligned}\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{u}_{t+1}-\mathbf{y}\|_2^2\right] \le \left(1-\frac{\theta\eta\lambda_0}{4}\right)^t \|\mathbf{u}_0-\mathbf{y}\|_2^2 \\ + O\Bigg(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2E^2}{n^2} + \frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4E^2}{m^4} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2 d}{m^3\lambda_0} \\ + \frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0E^2}{m^3} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2 d}{m^2\lambda_0} + \frac{n\kappa^2\left(\theta-\xi^2\right)}{S}\Bigg).\end{aligned}$$

We care about the MSE computed on the scaled full network:

$$u_k^{(i)} = \xi\sum_{r=1}^m\sum_{j=1}^p a_{rj}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r,t}\right\rangle\right); \quad \mathcal{L}\left(\mathbf{W}_t\right) = \|\mathbf{u}_t-\mathbf{y}\|_2^2.$$

Performing gradient descent on this scaled full network involves computing:

$$\nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_t\right) = \xi\sum_{i=1}^n\sum_{j=1}^p\left(u_t^{(i)}-y_i\right)a_{rj}\hat{\mathbf{x}}_i^{(j)}\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)};\mathbf{w}_{r,t}\right\}.$$

We will prove Theorem 4.1 by induction. We assume therom 4.1 is true for all $t' < t$ and $\|\mathbf{w}_{r,t'}-\mathbf{w}_{r,0}\|_2 \le R := O\left(\frac{\kappa\lambda_0}{n}\right)$.

4.1 Change of Activation Pattern

Let R be some fixed scale. For convenience, we denote:

$$A_{ir}^{(j)} = \left\{\exists\mathbf{w}:\|\mathbf{w}-\mathbf{w}_{r,0}\|_2\le R; \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)};\mathbf{w}\right\} \ne \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)};\mathbf{w}_{r,0}\right\}\right\}.$$

Note that $A_{ir}^{(j)}$ happens if and only if $\left|\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r,0}\right\rangle\right| < R$. Therefore $\mathbb{P}\left(A_{ir}^{(j)}\right) < \frac{2R}{\kappa\sqrt{2\pi}}$. Denote:

$$P_{ij} = \left\{r\in[m]:\neg A_{ir}^{(j)}\right\}; \quad P_{ij}^\perp = [m]\setminus P_{ij}.$$

The next lemma shows the magnitude of $P_{ij}^\perp$ is upper bounded by a controlled quantity.

Lemma 4.1 *Let $m = \Omega\left(R^{-1}\log\frac{np}{\delta}\right)$. Then with probability at least $1-O(\delta)$ it holds for all $i\in[n]$ and $j\in[p]$ that*

$$\left|P_{ij}^\perp\right| \le 3m\kappa^{-1}R.$$

Proof: The magnitude of $P_{ij}^\perp$ satisfies:

$$\left|P_{ij}^\perp\right| = \sum_{r=1}^m\mathbb{I}\left\{A_{ir}^{(j)}\right\}.$$

The indicator function $\mathbb{I}\left\{A_{ir}^{(j)}\right\}$ has bounded first and second moment:

$$\mathbb{E}_{\mathbf{W}}\left[\mathbb{I}\left\{A_{ir}^{(j)}\right\}\right] = \mathbb{P}\left(A_{ir}^{(j)}\right) \leq \frac{2R}{\kappa\sqrt{2\pi}},$$
$$\mathbb{E}_{\mathbf{W}}\left[\left(\mathbb{I}\left\{A_{ir}^{(j)} - \mathbb{E}_{\mathbf{W}}\left[\mathbb{I}\left\{A_{ir}^{(j)}\right\}\right]\right\}\right)^2\right] \leq \mathbb{E}_{\mathbf{W}}\left[\mathbb{I}\left\{A_{ir}^{(j)}\right\}^2\right] \leq \frac{2R}{\kappa\sqrt{2\pi}}.$$

This allows us to apply the Berstein Inequality to get that:

$$\mathbb{P}\left(\sum_{r=1}^{m}\mathbb{I}\left\{A_{ir}^{(j)}\right\} > \frac{2mR}{\kappa\sqrt{2\pi}} + mt\right) < \exp\left(-\frac{m\kappa t^2\sqrt{2\pi}}{8\left(1+\frac{t}{3}\right)R}\right).$$

Therefore, with probability at least $1 - np\exp\left(-m\kappa^{-1}R\right)$ it holds for all $i \in [n]$ and $j \in [p]$ that:

$$\left|P_{ij}^{\perp}\right| = \sum_{r=1}^{m}\mathbb{I}\left\{A_{ir}^{(j)}\right\} \leq 3m\kappa^{-1}R.$$

Letting $m = \Omega\left(R^{-1}\log\frac{np}{\delta}\right)$ gives that the success probability is at least $1 - O(\delta)$.

4.2 Initialization Scale

Let $\mathbf{w}_{0,r} \sim \mathcal{N}\left(0, \kappa^2\mathbf{I}\right)$ and $a_j \sim \left\{-\frac{1}{p\sqrt{m}}, \frac{1}{p\sqrt{m}}\right\}$ for all $r \in [m]$ and $j \in [p]$. The following lemmas hold true:

Lemma 4.2 *Suppose $\kappa \leq 1, R \leq \kappa\sqrt{\frac{d}{32}}$. With probability at least $1 - e^{md/32}$ we have that:*

$$\|W_0\|_F \leq \kappa\sqrt{2md} - \sqrt{m}R.$$

Lemma 4.3 *Assume $\kappa \leq 1$ and $R \leq \frac{\kappa}{\sqrt{2}}$. With probability at least $1 - ne^{-\frac{m}{32}}$ over initialization, it holds for all $i \in [n]$ that:*

$$\sum_{r=1}^{m}\left\langle\mathbf{w}_{0,r}, \mathbf{x}_i\right\rangle^2 \leq 2m\kappa^2 - mR^2$$
$$\sum_{i=1}^{n}\sum_{r=1}^{m}\left\langle\mathbf{w}_{0,r}, \mathbf{x}_i\right\rangle^2 \leq 2mn\kappa^2 - mnR^2.$$

Moreover, we can bound the initial MSE, based on the lemma below:

Lemma 4.4 *Assume that for all $i \in [n]$, y_i satisfies $|y_i| \leq C$ for some $C > 0$. Then, we have*

$$\mathbb{E}_{\mathbf{W}_0,\hat{\mathbf{a}}}\left[\|\mathbf{y} - \mathbf{u}_0\|_2^2\right] \leq \left(p^{-1} + C^2\right)n.$$

Proof: It is obvious that $\mathbb{E}_{\mathbf{W}_0,\hat{\mathbf{a}}}\left[u_0^{(i)}\right] = 0$ for all $i \in [n]$. Moreover:

$$\begin{aligned}
\mathbb{E}_{\mathbf{W}_0,\hat{\mathbf{a}}}\left[u_0^{(i)2}\right] &= \sum_{r,r'=1}^{m}\sum_{j,j'=1}^{p}\mathbb{E}_{\hat{\mathbf{a}}}\left[a_{rj}a_{r'j'}\right]\mathbb{E}_{\mathbf{W}_0}\left[\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{0,r}\right\rangle\right)\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j')}, \mathbf{w}_{0,r}\right\rangle\right)\right] \\
&= \frac{1}{p^2m}\sum_{r=1}^{m}\sum_{j=1}^{p}\mathbb{E}_{\mathbf{W}_0}\left[\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{0,r}\right\rangle\right)^2\right] \\
&\leq \frac{1}{p^2m}\sum_{r=1}^{m}\sum_{j=1}^{p}\mathbb{E}_{\mathbf{W}_0}\left[\left\langle\hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{0,r}\right\rangle\right] \\
&\leq p^{-1}
\end{aligned}$$

Therefore:

$$\mathbb{E}_{\mathbf{W}_0,\hat{\mathbf{a}}}\left[\|\mathbf{u}_0 - \mathbf{y}\|_2^2\right] = \sum_{i=1}^{n}\mathbb{E}_{\mathbf{W}_0,\hat{\mathbf{a}}}\left[\left(u_0^{(i)} - y_i\right)_2^2\right] = \sum_{i=1}^{n}\left(\mathbb{E}_{\mathbf{W}_0,\hat{\mathbf{a}}}\left[u_0^{(i)2}\right] + y_i^2\right) \leq \left(p^{-1} + C^2\right)n$$

4.3 Kernel Analysis

The neural tangent kernel is defined to be the inner product of the gradient with respect to the neural network output. We let the finite-width NTK be defined as:

$$\mathbf{H}(t)_{ii'} = \sum_{r=1}^{m} \sum_{j,j'=1}^{p} a_{rj} a_{rj'} \left\langle \hat{\mathbf{x}}_i^{(j)}, \hat{\mathbf{x}}_{i'}^{(j')} \right\rangle \mathbb{I}\left\{ \hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t} \right\} \mathbb{I}\left\{ \hat{\mathbf{x}}_{i'}^{(j')}; \mathbf{w}_{r,t} \right\}.$$

Moreover, let the infinite width NTK be defined as

$$\mathbf{H}_{ii'}^{\infty} = \frac{1}{p^2} \sum_{j=1}^{p} \left\langle \hat{\mathbf{x}}_i^{(j)}, \hat{\mathbf{x}}_{i'}^{(j)} \right\rangle \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(0,\mathbf{I})} \left[\mathbb{I}\left\{ \hat{\mathbf{x}}_i^{(j)}; \mathbf{w} \right\} \mathbb{I}\left\{ \hat{\mathbf{x}}_{i'}^{(j)}; \mathbf{w} \right\} \right].$$

Let $\lambda_0 = \lambda_{\min}\left(\mathbf{H}^{\infty}\right)$. Note that since $\hat{\mathbf{x}}_i \not\parallel \hat{\mathbf{x}}_{i'}$ for $i \neq i'$. Thus $\hat{\mathbf{x}}_i^{(j)} \not\parallel \hat{\mathbf{x}}_{i'}^{(j)}$ for $i \neq i'$. () shows that the matrix $\hat{\mathbf{H}}(j)$, as defined below, is positive definite for all $j \in [p]$:

$$\hat{\mathbf{H}}(j)_{ii'}^{\infty} = \left\langle \hat{\mathbf{x}}_i^{(j)}, \hat{\mathbf{x}}_{i'}^{(j)} \right\rangle \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(0,\mathbf{I})} \left[\mathbb{I}\left\{ \hat{\mathbf{x}}_i^{(j)}; \mathbf{w} \right\} \mathbb{I}\left\{ \hat{\mathbf{x}}_{i'}^{(j)}; \mathbf{w} \right\} \right].$$

Since $\mathbf{H}^{\infty} = p^{-2} \sum_{j=1}^{p} \hat{\mathbf{H}}(j)^{\infty}$, we have that $\mathbf{H}^{\infty}$ is positive definite and thus $\lambda_0 > 0$. The following lemma shows that the NTK remains positive definite throughout training.

Lemma 4.5 *Let $m = \Omega\left(\lambda_0^{-2} n^2 \log \frac{n}{\delta}\right)$. If for all $r \in [m]$ and all t we have $\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2 \leq R := O\left(\frac{\kappa \lambda_0}{n}\right)$. Then with probability at least $1 - \delta$ we have that for all t:*

$$\lambda_{\min}\left(\mathbf{H}(t)\right) \geq \frac{\lambda_0}{2}.$$

Proof: To start, we notice that for all $r \in [m]$:

$$\mathbb{E}_{\mathbf{W}_0, \mathbf{a}} \left[\sum_{j,j'=1}^{p} a_{rj} a_{rj'} \mathbb{I}\left\{ \hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,0} \right\} \mathbb{I}\left\{ \hat{\mathbf{x}}_{i'}^{(j')}; \mathbf{w}_{r,0} \right\} \right]$$
$$= \frac{1}{p^2 m} \mathbb{E}_{\mathbf{w} \sim \mathcal{N}(0,\mathbf{I})} \left[\mathbb{I}\left\{ \hat{\mathbf{x}}_i^{(j)}; \mathbf{w} \right\} \mathbb{I}\left\{ \hat{\mathbf{x}}_{i'}^{(j)}; \mathbf{w} \right\} \right]$$

Moreover, we have that:

$$\left| \sum_{j,j'=1}^{p} a_{rj} a_{rj'} \left\langle \hat{\mathbf{x}}_i^{(j)}, \hat{\mathbf{x}}_{i'}^{(j')} \right\rangle \mathbb{I}\left\{ \hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,0} \right\} \mathbb{I}\left\{ \hat{\mathbf{x}}_{i'}^{(j')}; \mathbf{w}_{r,0} \right\} \right| \leq 1.$$

Thus, we can apply Hoeffding's inequality with bounded random variable to get that

$$\mathbb{P}\left(\left| \mathbf{H}(0)_{i,i'} - \mathbf{H}_{i,i'}^{\infty} \right| \geq t \right) \leq 2 \exp\left(-mt^2\right).$$

Therefore, with probability at least $1 - O(\delta)$ it holds that for all $i, i' \in [n]$:

$$\left| \mathbf{H}(0)_{i,i'} - \mathbf{H}_{i,i'}^{\infty} \right| \leq \frac{\log \frac{n}{\delta}}{\sqrt{m}},$$

which implies that:

$$\|\mathbf{H}(0) - \mathbf{H}^{\infty}\| \leq \|\mathbf{H}(0) - \mathbf{H}^{\infty}\|_F \leq \frac{n\left(\log \delta^{-1} + \log n\right)}{\sqrt{m}}.$$

As long as $m = \Omega\left(\lambda_0^{-2} n^2 \log \frac{n}{\delta}\right)$, we will have:

$$\|\mathbf{H}(t) - \mathbf{H}^{\infty}\| \leq \frac{\lambda_0}{4}.$$

Now, we move on to bound $\|\mathbf{H}(t) - \mathbf{H}(0)\|$. We have that:

$$\mathbf{H}(t)_{i,i'} - \mathbf{H}(0)_{i,i'} = \sum_{r=1}^{m} \sum_{j,j'=1}^{p} a_{rj} a_{rj'} \left\langle \hat{\mathbf{x}}_i^{(j)}, \hat{\mathbf{x}}_{i'}^{(j')} \right\rangle z_{r,i,i'}^{(j,j')},$$

with

$$z_{r,i,i'}^{(j,j')} = \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \mathbb{I}\left\{\hat{\mathbf{x}}_{i'}^{(j')}; \mathbf{w}_{r,t}\right\} - \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,0}\right\} \mathbb{I}\left\{\hat{\mathbf{x}}_{i'}^{(j')}; \mathbf{w}_{r,0}\right\}.$$

We observe that $|z_{r,i,i'}^{(j,j')}|$ only if $A_{ir}^{(j)} \vee A_{i'r}^{(j')}$. Therefore:

$$\mathbb{E}_{\mathbf{W}}\left[z_{r,i,i'}^{(j,j')}\right] \leq \mathbb{P}\left(A_{ir}^{(j)}\right) + \mathbb{P}\left(A_{i'r}^{(j')}\right) \leq \frac{4R}{\kappa\sqrt{2\pi}}.$$

For the case $j = j'$, we first notice that

$$\mathbb{E}_{\mathbf{W}}\left[\left(z_{r,i,i'}^{(j,j')} - \mathbb{E}_{\mathbf{W}}\left[z_{r,i,i'}^{(j,j')}\right]\right)^2\right] \leq \mathbb{E}_{\mathbf{W}}\left[z_{r,i,i'}^{(j,j')2}\right] \leq \frac{4R}{\kappa\sqrt{2\pi}}.$$

Thus, applying Berstein Inequality to the case $j = j'$ we have that:

$$\mathbb{P}\left(\sum_{r=1}^{m} z_{r,i,i'}^{(j,j)} \geq m\left(\mathbb{E}_{\mathbf{W}}\left[z_{r,i,i'}^{(j,j')}\right] + t\right)\right) \leq \exp\left(-\frac{\kappa m t^2\sqrt{2\pi}}{8\left(1+\frac{t}{3}\right)R}\right).$$

For the case $j \neq j'$, we notice that:

$$\mathbb{E}_{\mathbf{W},\mathbf{a}}\left[a_{rj}a_{rj'}z_{r,i,i'}^{(j,j')}\right] = 0.$$

Moreover:

$$\left|a_{rj}a_{rj'}z_{r,i,i'}^{(j,j')}\right| \leq \frac{4R}{p^2 m\kappa\sqrt{2\pi}},$$
$$\mathbb{E}_{\mathbf{W},\mathbf{a}}\left[\left(a_{rj}a_{rj'}z_{r,i,i'}^{(j,j')}\right)^2\right] = \frac{1}{p^4m^2}\mathbb{E}_{\mathbf{W}}\left[z_{r,i,i'}^{(j,j')2}\right] \leq \frac{4R}{p^4m^2\kappa\sqrt{2\pi}}.$$

Applying the Berstein Inequality to the case $j \neq j'$, we have that:

$$\mathbb{P}\left(\sum_{r=1}^{m} a_{rj}a_{rj'}z_{r,i,i'}^{(j,j')} \geq \frac{t}{p^2}\right) \leq \exp\left(-\frac{m\kappa t^2\sqrt{2\pi}}{8\left(1+\frac{t}{3}\right)R}\right).$$

Combining both cases, we have that with probability at least $1 - p^2\exp\left(-\frac{m\kappa t^2\sqrt{2\pi}}{8\left(1+\frac{t}{3}\right)R}\right)$, it holds that:

$$|\mathbf{H}(t)_{i,i'} - \mathbf{H}(0)_{i,i'}| \leq p^{-1}\mathbb{E}_{\mathbf{W}}\left[z_{r,i,i'}^{(j,j')}\right] + t \leq \frac{2R}{p\kappa} + t^2.$$

Choose $t = \kappa^{-1}R$. Then as long as $m = \frac{\log\frac{np}{\delta}}{R}$, it holds that with probability at least $1 - O(\delta)$:

$$|\mathbf{H}(t)_{i,i'} - \mathbf{H}(0)_{i,i'}| \leq 3\kappa^{-1}R.$$

This implies that:

$$\|\mathbf{H}(t) - \mathbf{H}(0)\|_2 \leq \|\mathbf{H}(t) - \mathbf{H}(0)\|_F \leq 3n\kappa^{-1}R.$$

Thus, $\|\mathbf{H}(t) - \mathbf{H}(0)\|_2 \leq \frac{\lambda_0}{4}$ as long as $R = O\left(\frac{\kappa\lambda_0}{n}\right)$. This shows that $\lambda_{\min}\left(\mathbf{H}(t)\right) \geq \frac{\lambda_0}{2}$ for all t with probability at least $1 - O(\delta)$.

4.4 Surrogate Gradient Bound

As we see in previous section, the `AsyncDrop` scheme can be written as

$$\mathbf{w}_{r,t+1} = \mathbf{w}_{r,t} - \eta \mathbf{g}_{r,t-\delta_t}; \quad \mathbf{g}_{r,t} = \sum_{i=1}^{n}\sum_{j=1}^{p} a_{rj}\left(\tilde{u}_{r,t}^{(i)} - N_{r,t}^{\perp} y_i\right)\hat{\mathbf{x}}_i^{(j)}\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}$$

with $\tilde{u}_{r,t}^{(i)}$ defined as

$$\tilde{u}_{r,t}^{(i)} = \frac{N_{r,t}^{\perp}}{N_{r,t}}\sum_{s=1}^{S} m_{r,t}^{(s)}\hat{u}_t^{(s,i)} = \sum_{r'=1}^{m}\sum_{j=1}^{p}\underbrace{\left(\frac{N_{r,t}^{\perp}}{N_{r,t}}\sum_{s=1}^{S} m_{r,t}^{(s)} m_{r',t}^{(s)}\right)}_{\nu_{r,r',t}} a_{rj}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t}\right\rangle\right)$$

The mixing of the surrogate function $\tilde{u}_{r,t}^{(i)}$ can be bounded by

$$\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[\tilde{u}_{r,t}^{(i)}\right] &= \sum_{s=1}^{S}\sum_{r'=1}^{m}\sum_{j=1}^{p}\mathbb{E}_{\mathbf{M}_t}\left[m_{r,t}^{(s)} m_{r,t'}^{(s)}\cdot\frac{N_{r,t}^{\perp}}{N_{r,t}}\right] a_{r'j}\sigma\left(\left\langle\hat{\mathbf{x}}^{(j)}, \mathbf{w}_{r,t'}\right\rangle\right) \\
&= \xi\theta\sum_{r'=1}^{m}\sum_{j=1}^{p} a_{rj}\sigma\left(\left\langle\hat{\mathbf{x}}^{(j)}, \mathbf{w}_{r,t}\right\rangle\right) + (1-\xi)\theta\sum_{j=1}^{p} a_{rj}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t}\right\rangle\right) \\
&= \theta u_t^{(i)} + (1-\xi)\theta\underbrace{\sum_{j=1}^{p} a_{rj}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t}\right\rangle\right)}_{\hat{\epsilon}_{r,t}^{(i)}}
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[\mathbf{g}_{r,t}\right] &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{rj}\mathbb{E}_{\mathbf{M}_t}\left[\tilde{u}_t^{(i)} - N_{r,t}^{\perp} y_i\right]\hat{\mathbf{x}}_i^{(j)}\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&= \xi^{-1}\theta\nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_t\right) + (1-\xi)\theta\underbrace{\sum_{i=1}^{n}\sum_{j'=1}^{p} a_{rj'}\hat{\epsilon}_{r,t}^{(i)}\hat{\mathbf{x}}_i^{(j)}\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}}_{\boldsymbol{\epsilon}_{r,t}}
\end{aligned}$$

Now, we have

$$\left|\hat{\epsilon}_{r,t}^{(i)}\right| \leq \frac{1}{\sqrt{m}}\left\|\mathbf{w}_{r,t}\right\|_2; \quad \left\|\boldsymbol{\epsilon}_{r,t}\right\|_2 \leq \frac{n}{\sqrt{m}}\left|\hat{\epsilon}_{r,t}^{(i)}\right| \leq \frac{n}{m}\left\|\mathbf{w}_{r,t}\right\|_2$$

Moreover, we would like to investigate the norm and norm squared of the gradient. In particular, we first notice that, under the case of $N_{t,r}^{\perp} = 1$, we have

$$\mathbf{g}_{r,t} = \xi^{-1}\nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_t\right) + \sum_{i=1}^{n}\sum_{j=1}^{p} a_{rj}\left(\tilde{u}_{r,t}^{(i)} - u_t^{(i)}\right)\hat{\mathbf{x}}_i^{(j)}\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}$$

Thus, we are interested in $\left\|\tilde{\mathbf{u}}_{r,t} - \mathbf{u}_t\right\|_2$. Following from previous work [4] (lemma 19, 20, and 21), we have that

$$\mathbb{E}_{\mathbf{M}_t}\left[\nu_{r,r',t} \mid N_{r,t}^{\perp} = 1\right]\begin{cases}\xi & \text{if } r \neq r' \\ 1 & \text{if } r = r'\end{cases} \qquad \text{Var}\left(\nu_{r,r',t} \mid N_{r,t}^{\perp} = 1\right) = \begin{cases}\frac{\theta-\xi^2}{S} & \text{if } r \neq r' \\ 0 & \text{if } r = r'\end{cases}$$

Therefore

$$
\begin{aligned}
&\mathbb{E}_{\mathbf{M}_t}\left[\|\tilde{\mathbf{u}}_{r,t}-\mathbf{u}_t\|_2^2 \mid N_{r,t}^{\perp}=1\right] \\
&\quad=\sum_{i=1}^{n}\mathbb{E}_{\mathbf{M}_t}\left[\left(\sum_{r'=1}^{m}\sum_{j=1}^{p}\left(\nu_{r,r',t}-\xi\right)a_{rj}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r',t}\right\rangle\right)\right)^2 \mid N_{r,t}^{\perp}=1\right] \\
&\quad=\sum_{i=1}^{n}\sum_{r'=1}^{m}\mathbb{E}_{\mathbf{M}_t}\left[\left(\nu_{r,r',t}-\xi\right)^2 \mid N_{r,t}^{\perp}=1\right]\left(\sum_{j=1}^{p}a_{rj}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r',t}\right\rangle\right)\right)^2 \\
&\quad\le\frac{1}{mp}\sum_{i=1}^{n}\sum_{r'=1}^{m}\mathbb{E}_{\mathbf{M}_t}\left[\left(\nu_{r,r',t}-\xi\right)^2 \mid N_{r,t}^{\perp}=1\right]\sum_{j=1}^{p}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r',t}\right\rangle\right)^2 \\
&\quad\le\frac{1}{mp}\sum_{i=1}^{n}\sum_{r'\neq r}\mathrm{Var}\left(\nu_{r,r',t} \mid N_{r,t}^{\perp}=1\right)\sum_{j=1}^{p}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r',t}\right\rangle\right)^2+ \\
&\qquad\frac{1}{mp}\sum_{i=1}^{n}\mathbb{E}_{\mathbf{M}_t}\left[\left(\nu_{r,r,t}-\xi\right)^2 \mid N_{r,t}^{\perp}=1\right]\sum_{j=1}^{p}\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r',t}\right\rangle\right)^2 \\
&\quad\le\frac{1}{mpS}(\theta-\xi)\sum_{r'\neq r}\sum_{i=1}^{n}\sum_{j=1}^{p}\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r',t}\right\rangle^2+\frac{1}{mp}(\theta-\xi^2)\sum_{i=1}^{n}\sum_{j=1}^{p}\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r,t}\right\rangle^2
\end{aligned}
$$

With high probabilty it holds that

$$
\sum_{r=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{p}\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r,0}\right\rangle^2\le 2mnp\kappa^2-mnpR^2
$$

Thus, with sufficiently large m, the second term is always smaller than the first term, and we have

$$
\mathbb{E}_{\mathbf{M}_t}\left[\|\tilde{\mathbf{u}}_{r,t}-\mathbf{u}_t\|_2^2 \mid N_{r,t}^{\perp}=1\right]\le 8n\kappa^2(\theta-\xi^2)S^{-1}
$$

Now, we can compute that

$$
\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{g}_{r,t}\|_2 \mid N_{r,t}^{\perp}=1\right]&=\mathbb{E}_{\mathbf{M}_t}\left[\left\|\sum_{i=1}^{n}\sum_{j=1}^{p}a_{rj}\left(\tilde{u}_{r,t}^{(i)}-u_t^{(i)}\right)\hat{\mathbf{x}}_i^{(j)}\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)};\mathbf{w}_{r,t}\right\}\right\| \mid N_{r,t}=1\right]+ \\
&\qquad\xi^{-1}\theta\left\|\nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_t\right)\right\|_2 \\
&=\sqrt{\frac{n}{m}}\mathbb{E}_{\mathbf{M}_t}\left[\|\tilde{\mathbf{u}}_{r,t}-\mathbf{u}_t\|_2 \mid N_{r,t}=1\right]+\theta\sqrt{\frac{n}{m}}\|\mathbf{u}_t-\mathbf{y}\|_2 \\
&\le n\kappa\sqrt{\frac{\theta-\xi^2}{mS}}+\theta\sqrt{\frac{n}{m}}\|\mathbf{u}_t-\mathbf{y}\|_2
\end{aligned}
$$

And we know that $\mathbf{g}_{r,t}=0$ when $N_{r,t}^{\perp}=0$. Therefore,

$$
\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{g}_{r,t}\|_2\right]\le\theta n\kappa\sqrt{\frac{\theta-\xi^2}{mS}}+\theta^2\sqrt{\frac{n}{m}}\|\mathbf{u}_t-\mathbf{y}\|_2
$$

Similarly

$$
\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{g}_{r,t}\|_2^2 \mid N_{r,t}^{\perp}=1\right]&=\frac{2n}{m}\mathbb{E}_{\mathbf{M}_t}\left[\|\tilde{\mathbf{u}}_{r,t}-\mathbf{u}_t\|_2^2 \mid N_{r,t}=1\right]+\frac{2\theta^2n}{m}\|\mathbf{u}_t-\mathbf{y}\|_2^2 \\
&\le\frac{16n^2\kappa^2\left(\theta-\xi^2\right)}{mS}+\frac{2\theta^2n}{m}\|\mathbf{u}_t-\mathbf{y}\|_2^2
\end{aligned}
$$

and thus

$$
\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{g}_{r,t}\|_2^2\right]\le\frac{16\theta n^2\kappa^2\left(\theta-\xi^2\right)}{mS}+\frac{2\theta^3n}{m}\|\mathbf{u}_t-\mathbf{y}\|_2^2
$$

4.5 Step-wise Convergence

Consider

$$\begin{aligned}\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{u}_{t+1}-\mathbf{y}\|_2^2\right] &= \|\mathbf{u}_t-\mathbf{y}\|_2^2 - 2\langle \mathbf{u}_t-\mathbf{u}_{t+1}, \mathbf{u}_t-\mathbf{y}\rangle + \|\mathbf{u}_t-\mathbf{u}_{t+1}\|_2^2 \\ &= \|\mathbf{u}_t-\mathbf{y}\|_2^2 - 2\langle \mathbf{I}_{1,t}+\mathbf{I}_{2,t}, \mathbf{u}_t-\mathbf{y}\rangle + \|\mathbf{u}_t-\mathbf{u}_{t+1}\|_2^2 \\ &\le \|\mathbf{u}_t-\mathbf{y}\|_2^2 - 2\langle \mathbf{I}_{1,t}, \mathbf{u}_t-\mathbf{y}\rangle + 2\|\mathbf{I}_{2,t}\|_2\|\mathbf{u}_t-\mathbf{y}\|_2 + \|\mathbf{u}_t-\mathbf{u}_{t+1}\|_2^2\end{aligned}$$

Let $P_{ij} = \left\{r \in [m] : \neg A_{ir}^{(j)}\right\}$. Here $\mathbf{I}_{1,t}$ and $\mathbf{I}_{2,t}$ are characterized as in previous work.

$$\begin{aligned}I_{1,t}^{(i)} &= \sum_{j=1}^{p}\sum_{r\in P_{ij}} a_{rj}\left(\sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t}\right\rangle\right) - \sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t+1}\right\rangle\right)\right) \\ I_{2,t}^{(i)} &= \sum_{j=1}^{p}\sum_{r\in P_{ij}^{\perp}} a_{rj}\left(\sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t}\right\rangle\right) - \sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t+1}\right\rangle\right)\right)\end{aligned}$$

We first bound the magnitude of $\mathbf{I}_{2,t}$

$$\begin{aligned}\left|I_{2,t}^{(i)}\right| &= \frac{1}{p\sqrt{m}}\sum_{j=1}^{p}\sum_{r\in P_{ij}^{\perp}}\left|\left(\sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t}\right\rangle\right) - \sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t+1}\right\rangle\right)\right)\right| \\ &\le \frac{1}{p\sqrt{m}}\sum_{j=1}^{p}\sum_{r\in P_{ij}}\|\mathbf{w}_{r,t}-\mathbf{w}_{r,t+1}\|_2 \\ &\le \frac{\eta}{p\sqrt{m}}\sum_{j=1}^{p}\sum_{r\in P_{ij}}\|\mathbf{g}_{r,t-\delta_t}\|_2 \\ &\le \frac{\eta}{\sqrt{m}}\cdot 3m\kappa^{-1}R\cdot\left(n\kappa\sqrt{\frac{\theta-\xi^2}{mS}} + \theta\sqrt{\frac{n}{m}}\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\|_2\right) \\ &= 3\kappa^{-1}\theta\eta\sqrt{n}R\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\|_2 + 3\eta nR\sqrt{\frac{\theta-\xi^2}{S}}\end{aligned}$$

Thus

$$\begin{aligned}\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{I}_{2,t}\|_2\right] &\le \mathbb{E}_{\mathbf{M}_t}\left[\sqrt{n}\max_{i\in[n]}\left|I_{2,t}^{(i)}\right|\right] \\ &\le \frac{\eta}{p}\sqrt{\frac{n}{m}}\max_{i\in[n]}\sum_{j=1}^{p}\sum_{r\in P_{ij}}\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{g}_{r,t-\delta_t}\|_2\right] \\ &\le \eta\sqrt{\frac{n}{m}}\cdot 3m\kappa^{-1}R\cdot\left(n\kappa\theta\sqrt{\frac{\theta-\xi^2}{mS}} + \theta^2\sqrt{\frac{n}{m}}\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\|_2\right) \\ &= 3\kappa^{-1}\theta^2\eta nR\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\|_2 + 3\theta\eta n^{\frac{3}{2}}R\sqrt{\frac{\theta-\xi^2}{S}}\end{aligned}$$

Therefore, given assumption $\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\|_2 \ge \|\mathbf{u}_t-\mathbf{y}\|_2$

$$\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{I}_{2,t}\|_2\|\mathbf{u}_t-\mathbf{y}\|_2\right] \le 6\theta\eta n\kappa^{-1}R\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\|_2^2 + 3S^{-1}\theta\left(\theta-\xi^2\right)\eta n^2\kappa R$$

Letting $R = O\left(\frac{\kappa\lambda_0}{n}\right)$ gives that

$$\mathbb{E}_{\mathbf{M}_t}\left[\|\mathbf{I}_{2,t}\|_2\|\mathbf{u}_t-\mathbf{y}\|_2\right] = O\left(\theta\eta\lambda_0\right)\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\|_2^2 + O\left(\theta\eta\lambda_0(\theta-\xi^2)S^{-1}n\kappa^2\right)$$

As in previous work, $I_{1,t}^{(i)}$ can be written as

$$
\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[I_{1,t}^{(i)}\right] &= \xi \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \mathbb{E}_{\mathbf{M}_t}\left[\sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t} \right\rangle\right) - \sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t+1} \right\rangle\right)\right] \\
&= \xi \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbb{E}_{\mathbf{M}_t}\left[\mathbf{w}_{r,t} - \mathbf{w}_{r,t+1}\right] \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&= \xi\eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbb{E}_{\mathbf{M}_t}\left[\mathbf{g}_{r,t-\delta_t}\right] \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&= \xi\eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbb{E}_{\mathbf{M}_t}\left[\mathbf{g}_{r,t-\delta_t} - \mathbf{g}_{r,t} + \mathbf{g}_{r,t}\right] \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&= \xi\eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbb{E}_{\mathbf{M}_t}\left[\mathbf{g}_{r,t}\right] \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} + \\
&\qquad \xi\eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbb{E}_{\mathbf{M}_t}\left[\mathbf{g}_{r,t-\delta_t} - \mathbf{g}_{r,t}\right] \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&= \eta\theta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \nabla_{\mathbf{w}_r} \mathcal{L}\left(\mathbf{W}_t\right) \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&\qquad + \xi(1-\xi)\theta\eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \boldsymbol{\epsilon}_{r,t} \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&\qquad + \eta\theta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \nabla_{\mathbf{w}_r} \mathcal{L}\left(\mathbf{W}_{t-\delta_t}\right) - \nabla_{\mathbf{w}_r} \mathcal{L}\left(\mathbf{W}_t\right) \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&\qquad + \xi(1-\xi)\theta\eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \boldsymbol{\epsilon}_{r,t-\delta_t} - \boldsymbol{\epsilon}_{r,t} \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}
\end{aligned}
$$

$$
\begin{aligned}
&= \eta\theta\xi \sum_{i'=1}^{n} \sum_{j,j'=1}^{p} \sum_{r\in P_{ij}} \left(u_t^{(i')} - y_{i'}\right) a_{rj} a_{rj'} \left\langle \hat{\mathbf{x}}_i^{(j)}, \hat{\mathbf{x}}_{i'}^{(j')} \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \cdot \mathbb{I}\left\{\hat{\mathbf{x}}_{i'}^{(j')}; \mathbf{w}_{r,t}\right\} \\
&\quad + \xi(1-\xi)\theta\eta \sum_{j=1}^{p} \sum_{r\in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \boldsymbol{\epsilon}_{r,t} \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&\quad + \eta\theta \sum_{j=1}^{p} \sum_{r\in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_{t-\delta_t}\right) - \nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_t\right) \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&\quad + \xi(1-\xi)\theta\eta \sum_{j=1}^{p} \sum_{r\in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \boldsymbol{\epsilon}_{r,t-\delta_t} - \boldsymbol{\epsilon}_{r,t} \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \\
&= \eta\theta \sum_{i'=1}^{n} \left(\mathbf{H}(t)_{ii'} - \mathbf{H}(t)^{\perp}_{ii'}\right) \left(u_t^{(i')} - y_{i'}\right) \\
&\quad \underbrace{+ \xi(1-\xi)\theta\eta \sum_{j=1}^{p} \sum_{r\in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \boldsymbol{\epsilon}_{r,t} \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}}_{\gamma_{1,i,t}} \\
&\quad \underbrace{+ \eta\theta \sum_{j=1}^{p} \sum_{r\in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_{t-\delta_t}\right) - \nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_t\right) \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}}_{\gamma_{2,i,t}} \\
&\quad \underbrace{+ \xi(1-\xi)\theta\eta \sum_{j=1}^{p} \sum_{r\in P_{ij}} a_{rj} \left\langle \hat{\mathbf{x}}_i^{(j)}, \boldsymbol{\epsilon}_{r,t-\delta_t} - \boldsymbol{\epsilon}_{r,t} \right\rangle \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}}_{\gamma_{3,i,t}}
\end{aligned}
$$

Note that:

$$
\begin{aligned}
|\gamma_{1,i,t}| &\le \xi(1-\xi)\theta\eta m^{-\frac{1}{2}} \sum_{r=1}^{m} \|\boldsymbol{\epsilon}_{r,t}\| \\
&\le \xi(1-\xi)\theta\eta n m^{-1} \|\mathbf{W}_t\|_F \\
&\le O\left(\xi(1-\xi)\theta\eta n\kappa\sqrt{\frac{d}{m}}\right)
\end{aligned}
$$

Given that:

$$
\begin{aligned}
\left\|\left(u_{t-\delta_t}^{(i)} - u_t^{(i)}\right)\right\|_2 &= \left\| \xi \sum_{r=1}^{m}\sum_{j=1}^{p} a_{rj}\sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t-\delta_t} \right\rangle\right) - \xi \sum_{r=1}^{m}\sum_{j=1}^{p} a_{rj}\sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t} \right\rangle\right) \right\|_2 \\
&\le \xi \sum_{r=1}^{m}\sum_{j=1}^{p} \left\| a_{rj}\hat{\mathbf{x}}_i^{(j)} \right\|_2 \|\mathbf{w}_{r,t-\delta_t} - \mathbf{w}_{r,t}\|_2 \le \xi \frac{1}{\sqrt{m}} R_E
\end{aligned}
$$

We show:

$$
\begin{aligned}
&\|\nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_{t-\delta_t}\right) - \nabla_{\mathbf{w}_r}\mathcal{L}\left(\mathbf{W}_t\right)\|_2 \\
&\quad = \left\| \xi \sum_{i=1}^{n}\sum_{j=1}^{p} \left(u_{t-\delta_t}^{(i)} - y_i\right) a_{rj}\hat{\mathbf{x}}_i^{(j)} \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t-\delta_t}\right\} - \xi \sum_{i=1}^{n}\sum_{j=1}^{p} \left(u_t^{(i)} - y_i\right) a_{rj}\hat{\mathbf{x}}_i^{(j)} \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \right\|_2 \\
&\quad \le \xi \sum_{i=1}^{n}\sum_{j=1}^{p} \left\| a_{rj}\hat{\mathbf{x}}_i^{(j)} \right\|_2 \left\| \left(u_{t-\delta_t}^{(i)} - y_i\right) \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t-\delta_t}\right\} - \left(u_t^{(i)} - y_i\right) \mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\} \right\|_2 \\
&\quad \le \xi \sum_{i=1}^{n}\sum_{j=1}^{p} \left\| a_{rj}\hat{\mathbf{x}}_i^{(j)} \right\|_2 \left(\left\| \left(u_{t-\delta_t}^{(i)} - u_t^{(i)}\right) \right\|_2 + \left\| \left(u_t^{(i)} - y_i\right) \right\|_2 \right) \\
&\quad \le \xi^2 \frac{n}{m} R_E + \xi\sqrt{\frac{n}{m}} \|\mathbf{u}_t - \mathbf{y}\|_2
\end{aligned}
$$

Thus,

$$\|\gamma_{2,i,t}\|_2 \leq 2\eta\theta nm^{-2}\xi^2 R_E + \eta\theta\xi n^{\frac{1}{2}} m^{-\frac{3}{2}} \|\mathbf{u}_t - \mathbf{y}\|_2$$

Given that:

$$\begin{aligned}
\left\|\hat{\epsilon}^{(i)}_{r,t-\delta_t} - \hat{\epsilon}^{(i)}_{r,t}\right\|_2 &= \left\|\sum_{j=1}^{p} a_{rj}\sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t-\delta_t}\right\rangle\right) - \sum_{j=1}^{p} a_{rj}\sigma\left(\left\langle \hat{\mathbf{x}}_i^{(j)}, \mathbf{w}_{r,t}\right\rangle\right)\right\|_2 \\
&\leq \sum_{j=1}^{p} \left\|a_{rj}\hat{\mathbf{x}}_i^{(j)}\right\|_2 \left(\|\mathbf{w}_{r,t-\delta_t} - \mathbf{w}_{r,t}\|_2 + \|\mathbf{w}_{r,t}\|_2\right) \\
&\leq \kappa\sqrt{\frac{2d}{m}} + \frac{1}{\sqrt{m}} R_E
\end{aligned}$$

We show:

$$\begin{aligned}
\left\|\boldsymbol{\epsilon}^{(i)}_{r,t-\delta_t} - \boldsymbol{\epsilon}^{(i)}_{r,t}\right\|_2 &= \left\|\sum_{i=1}^{n}\sum_{j'=1}^{p} a_{rj'}\hat{\epsilon}^{(i)}_{r,t-\delta_t}\hat{\mathbf{x}}_i^{(j)}\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t-\delta_t}\right\} - \sum_{i=1}^{n}\sum_{j'=1}^{p} a_{rj'}\hat{\epsilon}^{(i)}_{r,t}\hat{\mathbf{x}}_i^{(j)}\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)}; \mathbf{w}_{r,t}\right\}\right\|_2 \\
&\leq \sum_{i=1}^{n}\sum_{j'=1}^{p} \left\|a_{rj'}\hat{\mathbf{x}}_i^{(j)}\right\|_2 \left(\left\|\left(\hat{\epsilon}^{(i)}_{r,t-\delta_t} - \hat{\epsilon}^{(i)}_{r,t}\right)\right\|_2 + \left\|\left(\hat{\epsilon}^{(i)}_{r,t}\right)\right\|_2\right) \\
&\leq \frac{n}{\sqrt{m}}\left(\kappa\sqrt{\frac{2d}{m}} + \frac{1}{\sqrt{m}}R_E + \frac{1}{\sqrt{m}}\kappa\sqrt{2dm}\right)
\end{aligned}$$

Thus,

$$\|\gamma_{3,i,t}\|_2 \leq \sqrt{2}\xi(1-\xi)\theta\eta\eta nm^{-\frac{3}{2}}\kappa\sqrt{d} + 2\xi(1-\xi)\theta\eta\eta nm^{-\frac{3}{2}}R_E + \sqrt{2}\xi(1-\xi)\theta\eta\eta nm^{-1}\kappa\sqrt{d})$$

This implies that:

$$\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[\langle \mathbf{I}_{1,t}, \mathbf{u}_t - \mathbf{y}\rangle\right] &= \eta\theta \sum_{i,i'=1}^{n}\left(u_t^{(i)} - y_i\right)\left(\mathbf{H}(t)_{ii'} - \mathbf{H}(t)^{\perp}_{ii'}\right)\left(u_t^{(i')} - y_{i'}\right) \\
&\quad + \sum_{i=1}^{n}\gamma_{1,i,t}\left(u_t^{(i)} - y_i\right) + \sum_{i=1}^{n}\gamma_{2,i,t}\left(u_t^{(i)} - y_i\right) + \sum_{i=1}^{n}\gamma_{3,i,t}\left(u_t^{(i)} - y_i\right) \\
&= \eta\theta\left\langle \mathbf{u}_t - \mathbf{y}, \left(\mathbf{H}(t) - \mathbf{H}(t)^{\perp}\right)(\mathbf{u}_t - \mathbf{y})\right\rangle \\
&\quad + \sum_{i=1}^{n}\gamma_{1,i,t}\left(u_t^{(i)} - y_i\right) + \sum_{i=1}^{n}\gamma_{2,i,t}\left(u_t^{(i)} - y_i\right) + \sum_{i=1}^{n}\gamma_{3,i,t}\left(u_t^{(i)} - y_i\right) \\
&\geq \eta\theta\left(\lambda_{\min}\left(\mathbf{H}(t)\right) - \lambda_{\max}\left(\mathbf{H}(t)^{\perp}\right)\right)\|\mathbf{u}_t - \mathbf{y}\|_2^2 \\
&\quad - \sum_{i=1}^{n}|\gamma_{1,i,t}| \cdot \left|u_t^{(i)} - y_i\right| - \sum_{i=1}^{n}|\gamma_{2,i,t}| \cdot \left|u_t^{(i)} - y_i\right| - \sum_{i=1}^{n}|\gamma_{3,i,t}| \cdot \left|u_t^{(i)} - y_i\right| \\
&\geq \eta\theta\left(\lambda_{\min}\left(\mathbf{H}(t)\right) - \lambda_{\max}\left(\mathbf{H}(t)^{\perp}\right)\right)\|\mathbf{u}_t - \mathbf{y}\|_2^2 \\
&\quad - \sqrt{n}\max_i|\gamma_{i,t}|\,\|\mathbf{u}_t - \mathbf{y}\|_2 \\
&\quad 2\eta\theta - n^{\frac{3}{2}}m^{-2}\xi^2 R_E\|\mathbf{u}_t - \mathbf{y}\|_2 - \eta\theta\xi nm^{-\frac{3}{2}}\|\mathbf{u}_t - \mathbf{y}\|_2^2 \\
&\quad - \sqrt{2}\xi(1-\xi)\theta\eta\eta nm^{-\frac{3}{2}}\kappa\sqrt{d}\,\|\mathbf{u}_t - \mathbf{y}\|_2 - 2\xi(1-\xi)\theta\eta\eta nm^{-\frac{3}{2}}R_E\|\mathbf{u}_t - \mathbf{y}\|_2 \\
&\quad \sqrt{2} - \xi(1-\xi)\theta\eta\eta nm^{-1}\kappa\sqrt{d}\,\|\mathbf{u}_t - \mathbf{y}\|_2 \\
&\geq \eta\theta\left(\lambda_{\min}\left(\mathbf{H}(t)\right) - \lambda_{\max}\left(\mathbf{H}(t)^{\perp}\right) - O\left(\lambda_0\right) - nm^{-\frac{3}{2}}\right)\|\mathbf{u}_t - \mathbf{y}\|_2^2 \\
&\quad - O\left(\frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n^3\xi^4 R_E^2}{m^4\lambda_0} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^3\lambda_0} +\right.
\end{aligned}$$

$$\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2R_E^2}{m^3\lambda_0}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^2\lambda_0}\Bigg)$$

For $\mathbf{H}(t)^\perp$ we have that

$$\begin{aligned}\lambda_{\max}\left(\mathbf{H}(t)^\perp\right)^2 &\leq \left\|\mathbf{H}(t)^\perp\right\|_F^2\\ &\leq \sum_{i,i'=1}^{n}\left(\sum_{j,j'=1}^{p}\sum_{r\in P_{ij}} a_{rj}a_{rj'}\left\langle\hat{\mathbf{x}}_i^{(j)},\hat{\mathbf{x}}_{i,i'}^{(j,j')}\right\rangle\mathbb{I}\left\{\hat{\mathbf{x}}_i^{(j)};\mathbf{w}_{r,t}\right\}\mathbb{I}\left\{\hat{\mathbf{x}}_{i,i'}^{(j,j')};\mathbf{w}_{r,t}\right\}\right)^2\\ &\leq \frac{n^2}{m^2}\left(\max_{ij}|P_{ij}|\right)^2\\ &\leq n^2\kappa^{-2}R^2\end{aligned}$$

Choosing $R=O\left(\frac{\kappa\lambda_0}{n}\right)$ gives

$$\lambda_{\max}\left(\mathbf{H}(t)^\perp\right)\leq O\left(\lambda_0\right)$$

We require $m\geq\Omega\left(\frac{n}{\lambda_0}\right)$, such that

$$nm^{-\frac{3}{2}}\leq O\left(\lambda_0\right)$$

Thus, Plugging in $\lambda_{\min}\left(\mathbf{H}(t)\right)\geq\frac{\lambda_0}{2}$, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{M}_t}\left[\langle\mathbf{I}_{1,t},\mathbf{u}_t-\mathbf{y}\rangle\right]\geq\eta\theta\lambda_0\left(\frac{1}{2}-O\left(1\right)\right)\left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2\\ -O\Bigg(\frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2d}{m\lambda_0}+\frac{\eta^2\theta^2n\kappa^2\lambda_0\xi^4E^2}{m^4}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0}+\\ \frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0E^2}{m^3}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^2\lambda_0}\Bigg).\end{aligned}$$

Lastly, we analyze the last term in the quadratic expansion

$$\begin{aligned}\mathbb{E}_{\mathbf{M}_t}\left[\left\|\mathbf{u}_t-\mathbf{u}_{t+1}\right\|_2^2\right]&=\sum_{i=1}^{n}\mathbb{E}_{\mathbf{M}_t}\left[\left(u_t^{(i)}-u_{t+1}^{(i)}\right)^2\right]\\ &\leq p^{-1}\sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{r=1}^{m}\mathbb{E}_{\mathbf{M}_t}\left[\left(\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r,t}\right\rangle\right)-\sigma\left(\left\langle\hat{\mathbf{x}}_i^{(j)},\mathbf{w}_{r,t+1}\right\rangle\right)\right)^2\right]\\ &\leq p^{-1}\eta^2\sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{r=1}^{m}\mathbb{E}_{\mathbf{M}_t}\left[\left\|\mathbf{g}_{r,t-\delta_t}\right\|_2^2\right]\\ &\leq O\left(\theta^3\eta^2n^2\right)\left\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\right\|_2^2+O\left(\theta\eta^2n^2\kappa^2\left(\theta-\xi^2\right)S^{-1}\right)\end{aligned}$$

Letting $\eta=O\left(\frac{\lambda_0}{n^2}\right)$ gives

$$\mathbb{E}_{\mathbf{M}_t}\left[\left\|\mathbf{u}_t-\mathbf{u}_{t+1}\right\|_2^2\right]\leq O\left(\theta\eta\lambda_0\right)\left\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\right\|_2^2+O\left(\theta\eta\lambda_0\kappa^2\left(\theta-\xi^2\right)S^{-1}\right)$$

Putting all three terms together and as $\left\|\mathbf{u}_{t-\delta_t}-\mathbf{y}\right\|_2^2\leq\left\|\mathbf{u}_{t-E}-\mathbf{y}\right\|_2^2$, we have that

$$
\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[\left\|\mathbf{u}_{t+1}-\mathbf{y}\right\|_2^2\right] &\leq \left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2-\eta\theta\lambda_0\left(1-O\left(1\right)\right)\left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2+O\left(\theta\eta\lambda_0\right)\left\|\mathbf{u}_{t-E}-\mathbf{y}\right\|_2^2 \\
&\quad+O\left(\frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0}+\frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4E^2}{m^4}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0}+\right. \\
&\qquad\left.\frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0E^2}{m^3}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^2\lambda_0}+\frac{n\kappa^2\left(\theta-\xi^2\right)}{S}\right) \\
&\leq \left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2-\eta\theta\lambda_0\left(1-O\left(1\right)\right)\left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2+O\left(\theta\eta\lambda_0\right)\left(2\left\|\mathbf{u}_{t-E}-\mathbf{u}_t\right\|_2^2+2\left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2\right) \\
&\quad+O\left(\frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0}+\frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4E^2}{m^4}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0}+\right. \\
&\qquad\left.\frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0E^2}{m^3}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^2\lambda_0}+\frac{n\kappa^2\left(\theta-\xi^2\right)}{S}\right) \\
&\leq \left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2-\eta\theta\lambda_0\left(1-O\left(1\right)\right)\left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2+O\left(\theta\eta\lambda_0\right)\left(8\xi^2R_E^2+2\left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2\right) \\
&\quad+O\left(\frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0}+\frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4E^2}{m^4}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0}+\right. \\
&\qquad\left.\frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0E^2}{m^3}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^2\lambda_0}+\frac{n\kappa^2\left(\theta-\xi^2\right)}{S}\right)
\end{aligned}
$$

Letting $O\left(\theta\eta\lambda_0\right)=\frac{1}{4}\theta\eta\lambda_0$,

$$
\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[\left\|\mathbf{u}_{t+1}-\mathbf{y}\right\|_2^2\right] &\leq \left(1-\frac{\theta\eta\lambda_0}{4}\right)\left\|\mathbf{u}_t-\mathbf{y}\right\|_2^2+ \\
&\quad O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2E^2}{n^2}+\frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0}+\frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4E^2}{m^4}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0}+\right. \\
&\qquad\left.\frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0E^2}{m^3}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^2\lambda_0}+\frac{n\kappa^2\left(\theta-\xi^2\right)}{S}\right)
\end{aligned}
$$

Thus,

$$
\begin{aligned}
\mathbb{E}_{\mathbf{M}_t}\left[\left\|\mathbf{u}_{t+1}-\mathbf{y}\right\|_2^2\right] &\leq \left(1-\frac{\theta\eta\lambda_0}{4}\right)^t\left\|\mathbf{u}_0-\mathbf{y}\right\|_2^2 \\
&\quad+O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2E^2}{n^2}+\frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0}+\frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4E^2}{m^4}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0}+\right. \\
&\qquad\left.\frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0E^2}{m^3}+\frac{\xi^2(1-\xi)^2\theta^2\eta^2n^2\kappa^2d}{m^2\lambda_0}+\frac{n\kappa^2\left(\theta-\xi^2\right)}{S}\right)
\end{aligned}
$$

4.6 Bounding Weight Perturbation

Next we show that $\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2 \leq R$ under sufficient over-parameterization. To start, we notice that

$$
\begin{aligned}
\mathbb{E}_{[\mathbf{M}_{t-1}]}[\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2] &\leq \sum_{t'=0}^{t-1} \mathbb{E}_{[\mathbf{M}_{t'}]} \left[\|\mathbf{w}_{r,t'+1} - \mathbf{w}_{r,t'}\|_2\right] \\
&\leq \eta \sum_{t'=0}^{t-1} \mathbb{E}_{[\mathbf{M}_{t'}]} \left[\left\|\mathbf{g}_{r,t'-\delta_{t'}}\right\|_2\right] \\
&\leq \eta \sum_{t'=0}^{t-1} \left(\theta^2 \sqrt{\frac{n}{m}} \mathbb{E}_{[\mathbf{M}_{t'-1}]} \left[\left\|\mathbf{u}_{t'-\delta_{t'}} - \mathbf{y}\right\|_2\right] + \theta n \kappa \sqrt{\frac{\theta - \xi^2}{mS}}\right) \\
&\leq \eta\theta^2 \sqrt{\frac{n}{m}} \sum_{t'=0}^{t-1} \mathbb{E}_{[\mathbf{M}_{t'-1}]} \left[\left\|\mathbf{u}_{t'-\delta_{t'}} - \mathbf{y}\right\|_2\right] + \eta t \theta n \kappa \sqrt{\frac{\theta - \xi^2}{mS}} \\
&\leq \eta\theta \sqrt{\frac{n}{m}} \|\mathbf{u}_0 - \mathbf{y}\|_2 \sum_{t'=0}^{t-1} \left(1 - \frac{\eta\theta\lambda_0}{4}\right)^{t'} + \eta T \theta n \kappa \sqrt{\frac{\theta - \xi^2}{mS}} \\
&\quad + \eta\theta T \cdot O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2}{n^2} + \frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4}{m^4} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^3\lambda_0} + \right. \\
&\qquad \left. \frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0}{m^3} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^2\lambda_0}\right) \\
&\leq \eta\theta \sqrt{\frac{n}{m}} \frac{1 - \left(1 - \frac{\eta\theta\lambda_0}{4}\right)^t}{\frac{\eta\theta\lambda_0}{4}} \|\mathbf{u}_0 - \mathbf{y}\|_2 \\
&\quad + \eta\theta T \cdot O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2}{n^2} + \frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4}{m^4} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^3\lambda_0} + \right. \\
&\qquad \left. \frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0}{m^3} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^2\lambda_0}\right) \\
&\leq \eta\theta \sqrt{\frac{n}{m}} \frac{1 - \left(1 - \frac{\eta\theta\lambda_0 t}{4}\right)}{\frac{\eta\theta\lambda_0}{4}} \|\mathbf{u}_0 - \mathbf{y}\|_2 \\
&\quad + \eta\theta T \cdot O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2}{n^2} + \frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4}{m^4} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^3\lambda_0} + \right. \\
&\qquad \left. \frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0}{m^3} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^2\lambda_0}\right) \\
&\leq O\left(\eta\theta\lambda_0 t \sqrt{\frac{n}{m}}\right) \|\mathbf{u}_0 - \mathbf{y}\|_2 \\
&\quad + \eta\theta T \cdot O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2}{n^2} + \frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4}{m^4} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^3\lambda_0} + \right. \\
&\qquad \left. \frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0}{m^3} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^2\lambda_0}\right)
\end{aligned}
$$

where the last inequality follows from the geometric sum and $\beta \leq O\left(p^{-1}\right)$. Using the initialization scale, we have that

$$
\begin{aligned}
\mathbb{E}_{\mathbf{W},\mathbf{a},[\mathbf{M}_t]} \left[\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2\right] \leq O\left(\eta\theta\lambda_0 t n m^{-\frac{1}{2}}\right) + \eta\theta T \cdot O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2}{n^2} + \frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4}{m^4} + \right. \\
\left. \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^3\lambda_0} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0}{m^3} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^2\lambda_0}\right)
\end{aligned}
$$

With probability $1-\delta$, it holds for all $t \in [T]$ that

$$\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2 \leq O\left(\frac{t\lambda_0^2 K}{\delta n\sqrt{m}}\right) + \eta\theta T \cdot O\left(\frac{\theta\eta\lambda_0^3\xi^2\kappa^2}{n^2} + \frac{\xi^2(1-\xi)^2\theta\eta n^3\kappa^2 d}{m\lambda_0} + \frac{\eta^2\theta^2 n\kappa^2\lambda_0\xi^4}{m^4} + \right.$$
$$\left. \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^3\lambda_0} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2\kappa^2\lambda_0}{m^3} + \frac{\xi^2(1-\xi)^2\theta^2\eta^2 n^2\kappa^2 d}{m^2\lambda_0}\right)$$

To enforce $\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2 \leq R := O\left(\frac{\kappa\lambda_0}{n}\right)$, we then require

$$m = \Omega\left(\frac{n^3 K^2}{\lambda_0^4\delta^2\kappa^2}\max\{n, d\}\right)$$

More specific,

$$\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2 \leq R_t := O\left(\frac{t\kappa\lambda_0}{n}\right)$$
$$\|\mathbf{w}_{r,t} - \mathbf{w}_{r,t-E}\|_2 \leq R_E := O\left(\frac{E\kappa\lambda_0}{n}\right)$$

References

[1] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019. [Cited on page 5.]

[2] Chen Dun, Cameron R Wolfe, Christopher M Jermaine, and Anastasios Kyrillidis. ResIST: Layer-wise decomposition of ResNets for distributed training. In *Uncertainty in Artificial Intelligence*, pages 610–620. PMLR, 2022. [Cited on pages 2 and 3.]

[3] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. [Cited on page 3.]

[4] Fangshuo Liao and Anastasios Kyrillidis. On the convergence of shallow neural network training with randomly masked neurons, 2021. [Cited on page 10.]

[5] Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018. [Cited on page 2.]