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 an 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
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, 45, 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 [47] [43], 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 [13], 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 $\ell(w, \cdot)$ over a dataset, such that the model maps unseen data to their true labels, unless otherwise stated. For clarity, the loss $\ell(w, \cdot)$ encodes both the loss metric and the neural architecture, with parameters $w$. Formally, given a data distribution $D$ and $\{x_i, y_i\} \sim D$, where $x_i$ is a data sample, and $y_i$ is its corresponding label, classical deep learning aims in finding $w^*$ as in:

$$w^* = \arg\min_{w \in \mathcal{H}} \left\{ \mathcal{L}(w) := \frac{1}{n} \sum_{i=1}^{n} \ell(w, \{x_i, y_i\}) \right\},$$

where $\mathcal{H}$ denotes the model hypothesis class that “molds” the trainable parameters $w$.

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 $w_t$ using stochastic directions $\nabla \ell_i(w_t) := \nabla \ell(w_t, \{x_i, y_i\})$. I.e., $w_{t+1} \leftarrow w_t - \eta \nabla \ell_i(w_t)$. Here, $\eta > 0$ is the learning rate, and $\ell_i$ 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].
FL formulation. Let $S$ be the total number of clients in a distributed FL scenario. Each client $i$ has its own local data $D_i$ such that the whole dataset satisfies $D = \bigcup_{i} D_i$, and usually $D_i \cap D_j = \emptyset, \forall i \neq j$. The goal of FL is to find a global model $W$ that achieves good accuracy on all data $D$, by minimizing the following optimization problem:

$$W^* = \arg\min_{W \in \mathcal{H}} \left\{ \mathcal{L}(W) := \frac{1}{S} \sum_{i=1}^{S} \ell(W, D_i) \right\},$$

where $\ell(W, D_i) = \frac{1}{|D_i|} \sum_{(x_j, y_j) \in D_i} \ell(W, \{x_j, y_j\})$. With a slight abuse of notation, $\ell(W, D_i)$ denotes the local loss function for user $i$, associated with a local model $W_i$ (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 $D_i$ can be heterogeneous and follow a non-i.i.d. distribution.

Algorithm 1 Meta Asynchronous FL

**Parameters:** $T$ iters, $S$ clients, $l$ local iters., $W$ as current global model, $W_i$ as local model for $i$-th client, $\alpha \in (0, 1)$, $\eta$ step size.

```plaintext
W ← randomly initialized global model.
//On each client i asynchronously:
for $t = 0, \ldots, T - 1$ do
    $W_{i,t} ← W$
    //Train $W_i$ for $l$ iters. via SGD
    for $j = 1, \ldots, l$ do
        $W_{i,t} ← W_{i,t} - \eta \frac{\partial \mathcal{L}}{\partial W_{i,t}}$
    end for
    //Write local to global model
    $W ← (1 - \alpha) \cdot W + \alpha \cdot W_{i,t}$
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 $W_i$ from the server, and further locally trains it using $D_i$ for a number of local iterations $l$. 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.

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.

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-i.i.d.” 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 [4, 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, $I$ local iters., $W$ as current global model, $W_i$ as local model for $i$-th client, $\alpha \in (0, 1)$, $\eta$ step size.

\begin{align*}
W & \leftarrow \text{randomly initialized global model.} \\
//\text{On each client } i \text{ asynchronously:} & \\
\text{for } t = 0, \ldots, T - 1 \text{ do} & \\
& \quad \text{Generate mask } M_{i,t} \\
& \quad W_{i,t} \leftarrow W_t \odot M_{i,t} \\
& \quad //\text{Train } W_{i,t} \text{ for } I \text{ iters. via SGD} \\
& \quad \text{for } j = 1, \ldots, I \text{ do} & \\
& \quad \quad W_{i,t} \leftarrow W_{i,t} - \eta \frac{\partial c}{\partial W_{i,t}} \\
& \quad \text{end for} & \\
& \quad //\text{Write local to global model} & \\
& \quad W_{t+1} \leftarrow W_t \odot (M_{t,I})^C & \\
& \quad \quad + ((1 - \alpha) \cdot W_t + \alpha \cdot W_{i,t}) \odot M_{i,t} \\
& \quad \text{end for} & \\
\end{align*}

Our proposal and main hypothesis. We focus on the asynchronous version of distributed dropout. We study theoretically whether asynchrony provably works on 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 $(X, y) = \{(x_i, y_i)\}_{i=1}^n$, where each $x_i \in \mathbb{R}^{d \times p}$ is an image and $y_i$ being its label. Here, $d$ 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 $\hat{\phi}(x) \in \mathbb{R}^{qd \times p}$. Consider the first layer weight $W \in \mathbb{R}^{m \times qd}$, and second layer (aggregation) weight $a \in \mathbb{R}^{m \times p}$. We assume that only the first layer weights $W$ is trainable. The CNN trained on the means squared error has the form:

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

where $f(x, \cdot)$ denotes the output of the one-layer CNN for input $x$, and $L(\cdot)$ is the loss function. We use the $\ell_2$-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 $\|x_i\|_F = q^{-\frac{1}{2}}$ and $|y_i| \leq C$ for some constant $C$. Moreover, for all $i, i' \in [n]$ we have $x_i \neq x_{i'}$.

Note that this can be satisfied by normalizing the data. For simplicity of the analysis, let $d := qd$.

Assumption 4.2 (Initialization) $w_{0,i} \sim \mathcal{N}(0, \kappa^2 \mathbf{I})$ and $a_{i,i'} \sim \left\{ \pm \frac{1}{\sqrt{pq}} \right\}$ 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 $\delta_t$ is the delay at training step $t$. We assume $\delta_t$ is bounded by a constant $E$. Then, a simple version of gradient descent under these assumptions looks like:

$$W_t = W_{t - \delta_t}, \quad \delta_t \leq E,$$

where $W_{t - \delta_t}$ 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 $u_t$ 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 $\xi$ 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 $m = \Omega\left(\max\{n^k \frac{\kappa^2}{m} \max\{n, d\}, \frac{n}{\lambda} \right\}$ and the step size satisfies $\eta = O\left(\frac{\lambda}{n^2}\right)$. Let $\kappa$ be a proper initialization scaling factor, and it is considered constant. We use $\lambda_0$ 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:

$$\mathbb{E}_{M_i}\left[\|u_{t+1} - y\|^2\right] \leq \left(1 - \frac{\theta \eta \lambda_0}{4}\right)^t \|u_0 - y\|^2 + O\left(\frac{\theta \eta \lambda_0 \xi^2 \kappa^2 E^2}{n^2} + \frac{\xi^2(1 - \xi)^2 \theta \eta n^3 \kappa^2 d}{m \lambda_0} + \frac{\eta^2 \theta^2 m \kappa^2 \lambda_0 \xi^4 E^2}{m^4} + \frac{\xi^2(1 - \xi)^2 \theta^2 \eta^2 n^3 \kappa^2 d}{m^3 \lambda_0} + \frac{\xi^2(1 - \xi)^2 \theta^2 \eta^2 n^3 \kappa^2 d}{m^2 \lambda_0} + \frac{n \kappa^2 (\theta - \xi^2)}{S}\right)$$

**Remark #1.** This theorem states that the sum of the expected weight differences in the $t$-th iteration (i.e., $\mathbb{E}_{M_i}\left[\|u_{t+1} - y\|^2\right]$) converges linearly to zero, as dictated by the red term $\left(1 - \frac{\theta \eta \lambda_0}{4}\right)^t \|u_0 - y\|^2$ 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: 

1. 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).

2. 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 $\frac{\theta \eta \lambda_0 \xi^2 \kappa^2 E^2}{n^2}$ 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...
Running heading title breaks the line

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.

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

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, 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 slower 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 in set \( \mathcal{J} \), define \( W_i^j \) 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: \( v(W_i^j) = \|W_i^j - W_i^0\|_1 \), 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 \( \varphi(W_i, \psi(i), v(\cdot)) \) generates masks that nullify weights in \( W_i \), based on the score function \( v \) over the
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.

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

### 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.

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

<table>
<thead>
<tr>
<th>CIFAR100</th>
<th>Max. Test Accuracy</th>
<th>Time for 32% Accuracy</th>
<th>Time Overhead</th>
<th>Comm. Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Async. FedAvg</td>
<td>32.47 ± 1.89</td>
<td>3062.5s</td>
<td>+33.38%</td>
<td>+15.56%</td>
</tr>
<tr>
<td>Async. Fed-Weighted-Avg</td>
<td>32.98 ± 1.71</td>
<td>3062.5s</td>
<td>+33.38%</td>
<td>+15.56%</td>
</tr>
<tr>
<td>Async. FedProx</td>
<td>35.75 ± 0.61</td>
<td>3062.5s</td>
<td>+33.38%</td>
<td>+15.56%</td>
</tr>
<tr>
<td>Async. FjORD</td>
<td>12.07 ± 0.83</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FedBuff</td>
<td>41.91 ± 3.80</td>
<td>4250.1s</td>
<td>+85.03%</td>
<td>+42.22%</td>
</tr>
<tr>
<td>Hetero AsyncDrop</td>
<td>37.26 ± 0.93</td>
<td>2296.8s</td>
<td>[Best]</td>
<td>[Best]</td>
</tr>
<tr>
<td>AsyncDrop</td>
<td>35.93 ± 0.93</td>
<td>2296.8s</td>
<td>[Best]</td>
<td>[Best]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FMNIST</th>
<th>Max. Test Accuracy</th>
<th>Time for 57% Accuracy</th>
<th>Time Overhead</th>
<th>Comm. Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Async. FedAvg</td>
<td>62.47 ± 8.20</td>
<td>787.5s</td>
<td>+33.33%</td>
<td>+25.00%</td>
</tr>
<tr>
<td>Async. Fed-Weighted-Avg</td>
<td>59.30 ± 10.44</td>
<td>1181.3s</td>
<td>+100%</td>
<td>+87.50%</td>
</tr>
<tr>
<td>Async. FedProx</td>
<td>59.91 ± 7.32</td>
<td>1050.0s</td>
<td>+77.78%</td>
<td>+66.67%</td>
</tr>
<tr>
<td>Async. FjORD</td>
<td>21.98 ± 9.55</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FedBuff</td>
<td>57.03 ± 11.37</td>
<td>1845.0s</td>
<td>+212.38%</td>
<td>+150.20%</td>
</tr>
<tr>
<td>Hetero AsyncDrop</td>
<td>66.89 ± 5.36</td>
<td>590.6s</td>
<td>[Best]</td>
<td>[Best]</td>
</tr>
<tr>
<td>AsyncDrop</td>
<td>60.02 ± 10.38</td>
<td>787.5s</td>
<td>+33.33%</td>
<td>+33.33%</td>
</tr>
</tbody>
</table>
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 overfitting [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 in the Appendix, 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


[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 4 and 6.]


1 Experimental Results using \textit{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.

<table>
<thead>
<tr>
<th>Max. Test Accuracy</th>
<th>Time for 59.95% Accuracy</th>
<th>Time Overhead</th>
<th>Comm. Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Async. FedAvg</td>
<td>61.59 ± 10.02</td>
<td>1096.1s</td>
<td>+33.33%</td>
</tr>
<tr>
<td>Async. Fed-Weighted-Avg</td>
<td>60.14 ± 10.28</td>
<td>1096.1s</td>
<td>+33.33%</td>
</tr>
<tr>
<td>Async. FedProx</td>
<td>61.25 ± 9.54</td>
<td>1096.1s</td>
<td>+33.33%</td>
</tr>
<tr>
<td>Async. FjORD</td>
<td>36.92 ± 6.00</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FedBuff</td>
<td>59.96 ± 7.64</td>
<td>1827.3s</td>
<td>+166.50%</td>
</tr>
<tr>
<td>Hetero AsyncDrop</td>
<td>61.90 ± 6.39</td>
<td>822.5s</td>
<td>[Best]</td>
</tr>
<tr>
<td>AsyncDrop</td>
<td>60.85 ± 9.78</td>
<td>822.5s</td>
<td>[Best]</td>
</tr>
</tbody>
</table>

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.

![Figure 1: Left: MLP FMNIST non-i.i.d.; Right: LSTM IMDB, non-i.i.d.](image)
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.

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

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.

<table>
<thead>
<tr>
<th>Dropout Ratio</th>
<th>Test Accuracy</th>
<th>Time for Accuracy to 23%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>32.47 ± 1.89</td>
<td>2105.4s</td>
</tr>
<tr>
<td>25%</td>
<td>37.26 ± 0.93</td>
<td>1579.1s</td>
</tr>
<tr>
<td>50%</td>
<td>23.95 ± 1.07</td>
<td>1531.2s</td>
</tr>
</tbody>
</table>

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. 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](image)

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

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 AsyncDrop using layerwise Smart Partition 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 experiments. As shown in Figure 3, Layerwise Hetero AsyncDrop results in a non-negligible decrease in performance, with respect to the final accuracy. This might show layer dropping causes increased variance and instability, especially in asynchronous training.

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.
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 permanently going offline.

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

3 Detailed Mathematical Formulation of AsyncDrop

For a vector \( \mathbf{v} \), \( \| \mathbf{v} \|_2 \) denotes its Euclidean (\( \ell_2 \)) 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 \( \mathbf{1} \{ \cdot \} \) to denote the indicator function. For two vectors \( \mathbf{v}_1, \mathbf{v}_2 \), we use the simplified notation \( \mathbf{1} \{ \mathbf{v}_1; \mathbf{v}_2 \} := \mathbf{1} \{ \langle \mathbf{v}_1, \mathbf{v}_2 \rangle \geq 0 \} \). Given that the mask in iteration \( t \) is \( M_t \), we denote \( \mathbb{E}[M_t] \left[ \cdot \right] = \mathbb{E}_{M_0,...,M_t} \left[ \cdot \right] \).

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

\[
f(\mathbf{x}, \theta) = \langle \mathbf{a}, \sigma (\mathbf{W} \hat{\phi}(\mathbf{x})) \rangle.
\]

Denote \( \hat{x} = \hat{\phi}(\mathbf{x}) \). 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 \( \hat{x}^{(j)}_i \in \mathbb{R}^{q\cdot d} \) as the extended \( j \)th pixel across all channels in the \( i \)th sample. For each transformed sample, we have that \( \| \hat{x}_i \|_F \leq \sqrt{q} \| x_i \|_F \). We simplify the CNN output as:

\[
f(\hat{x}, \mathbf{W}) = \langle \mathbf{a}, \sigma (\mathbf{W} \odot \hat{x}) \rangle = \sum_{r=1}^{m_1} \sum_{j=1}^{p} \alpha_{rj} \sigma \left( \langle \hat{x}^{(j)}_i, w_r \rangle \right).
\]

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

\[
\mathcal{L} (\mathbf{W}) = \left\| f \left( \hat{\mathbf{X}}, \mathbf{W} \right) - \mathbf{y} \right\|_2^2 = \sum_{i=1}^{n} (f (\hat{x}_i, \mathbf{W}) - y_i)^2.
\]
Similar to HogWild! training is fully lock free and thus cannot prevent several clients from updating the global model at the 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:

\[
    f_{m^{(s)}}(\hat{x}, W) = \sum_{r=1}^{m_1} \sum_{j=1}^{p} m_{i}^{(s)} a_{rj} \sigma \left( \langle \hat{x}^{(j)}, w_r \rangle \right).
\]

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

\[
    \nabla_{w_r} \mathcal{L}_{m^{(s)}} (W) = m_r^{(s)} \sum_{i=1}^{n} \sum_{j=1}^{p} (f_{m^{(s)}}(\hat{x}_i, W) - y_i) a_{rj} \hat{x}_i^{(j)} \parallel \{ x_i^{(j)}; w_r \} .
\]

We correspondingly scale the whole network function:

\[
    f(\hat{x}, W) = \sum_{r=1}^{m_1} \sum_{j=1}^{p} a_{rj} \sigma \left( \langle \hat{x}^{(j)}, w_r \rangle \right).
\]

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

\[
    \nabla_{w_r} \mathcal{L} (W) = \sum_{i=1}^{n} \sum_{j=1}^{p} \sum_{j=1}^{p} (f(\hat{x}_i, W) - y_i) a_{rj} \hat{x}_i^{(j)} \parallel \{ x_i^{(j)}; w_r \} .
\]

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

\[
    w_{r,t+1} = w_r - \eta \frac{N_{r,t}^{1}}{N_{r,t}^{1}-\delta_t} \sum_{s=1}^{S} \nabla_{w_r} \mathcal{L}_{m^{(s)}} (W_{r,t-\delta_t}).
\]

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

\[
    \bar{u}_{r,t} = \frac{N_{r,t}^{-1}}{N_{r,t}} \sum_{s=1}^{S} m_{r,t}^{(s)} u_t^{(s,i)}; \quad g_{r,t} = \frac{N_{r,t}^{-1}}{N_{r,t}} \sum_{s=1}^{S} \nabla_{w_r} \mathcal{L}_{m^{(s)}} (W_t).
\]

Then the AsyncDrop training has the form

\[
    w_{r,t+1} = w_r - \eta g_{r,t-\delta_t}; \quad g_{r,t} = \sum_{i=1}^{n} \sum_{j=1}^{p} a_{rj} \left( \bar{u}_{r,t} - N_{r,t}^{-1} y_t \right) \hat{x}_i^{(j)} \parallel \{ x_i^{(j)}; w_r \} .
\]

Suppose that Assumptions 1 and 2 in main text hold. Then for all \( i \in [n] \) we have \( \| x_i \|_F = q \sqrt{2} \), and for all \( i, i' \in [n] \) such that \( i \neq i' \), we have \( x_i \neq x_{i'} \). As in previous work \( \Pi \), we have \( \| \hat{x}_i \|_F = \leq \sqrt{q} \| x_i \|_F \leq 1 \). Thus, for all \( j \in [p] \) we have \( \| \hat{x}_i^{(j)} \|_F \leq 1 \). Moreover, since \( x_i \neq x_{i'} \) for \( i \neq i' \), we then have \( \hat{x}_i \neq \hat{x}_{i'} \), which implies that \( \hat{x}_i^{(j)} \neq \hat{x}_{i'}^{(j)} \) 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 $u_t$ abstractly represent the output of the model after $t$ iterations, over the random selection of the masks. Let $E$ denote the maximum gradient delay/staleness. Let $\xi$ denote the dropout rate ($\xi = 1$ dictates that all neurons are selected), and denote $\theta = 1 - (1 - \xi)^2$ the probability that a neuron is active in at least one subnetwork. Assume the number of hidden neurons satisfies $m = \Omega \left( \max \{ \frac{\kappa^2}{\delta^2}, \max\{n, d\}, \frac{\kappa}{\lambda_0} \} \right)$ and the step size satisfies $\eta = O \left( \frac{\kappa}{\delta} \right)$. Let $\kappa$ be a proper initialization scaling factor, and it is considered constant. We use $\lambda_0$ 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:

$$\mathbb{E}_{x_t} \left[ \left\| u_{t+1} - y \right\|_2 \right] \leq \left( 1 - \frac{\theta \eta \lambda_0}{4} \right)^t \left\| u_0 - y \right\|_2 + O \left( \frac{\theta \eta \lambda_0^3 \xi^2 \kappa^2 E^2}{m^2} + \frac{\eta^2 \theta^2 \eta \kappa^2 \lambda_0^4 E^2}{m^4} + \frac{\xi^2 (1 - \xi)^2 \theta^2 \eta \kappa^2 \lambda_0^2 E^2}{m^3 \lambda_0} + \frac{\xi^2 (1 - \xi)^2 \theta^2 \eta \kappa^2 \lambda_0^4 E^2}{m^4} + \frac{\eta \kappa^2 (\theta - \xi^2)^2}{S} \right).$$

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

$$u^{(i)}_k = \xi \sum_{r=1}^{m} \sum_{j=1}^{p} a_{rj} \sigma \left( \left\langle \hat{x}^{(j)}_i, w_{r,t} \right\rangle \right); \quad \mathcal{L}(W_t) = \left\| u_t - y \right\|_2^2.$$

Performing gradient descent on this scaled full network involves computing:

$$\nabla_{w_t} \mathcal{L}(W_t) = \xi \sum_{i=1}^{n} \sum_{j=1}^{p} \left( u^{(i)}_t - y_i \right) a_{rj} \hat{x}^{(j)}_i \mathbb{I} \left\{ \hat{x}^{(j)}_i; w_{r,t} \right\}.$$

We will prove Theorem 4.1 by induction. We assume theorem 4.1 is true for all $t' < t$ and $\|w_{r,t'} - w_{r,0}\|_2 \leq R := O \left( \frac{\sqrt{\lambda_0}}{m} \right)$.

4.1 Change of Activation Pattern

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

$$A_{ir}^{(j)} = \left\{ w : \| w - w_{r,0} \|_2 \leq R; \mathbb{I} \left\{ \hat{x}^{(j)}_i; w \right\} \neq \mathbb{I} \left\{ \hat{x}^{(j)}_i; w_{r,0} \right\} \right\}.$$

Note that $A_{ir}^{(j)}$ happens if and only if $\left\langle \hat{x}^{(j)}_i, w_{r,0} \right\rangle < R$. Therefore $\mathbb{P}(A_{ir}^{(j)}) < \frac{2R}{\kappa \sqrt{2\pi}}$. 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 $m = \Omega \left( R^{-1} \log \frac{m^2}{\delta} \right)$. Then with probability at least $1 - O(\delta)$ it holds for all $i \in [n]$ and $j \in [p]$ that

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

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

$$|P_{ij}^\perp| = \sum_{r=1}^{m} \mathbb{I} \left\{ A_{ir}^{(j)} \right\}.$$
The indicator function $\mathbb{I}\left\{ A_{ir}^{(j)} \right\}$ has bounded first and second moment:

\[
\mathbb{E}_W \left[ \mathbb{I} \left\{ A_{ir}^{(j)} \right\} \right] = \mathbb{P} \left( A_{ir}^{(j)} \right) \leq \frac{2R}{\kappa \sqrt{2\pi}},
\]

\[
\mathbb{E}_W \left[ \left( \mathbb{I} \left\{ A_{ir}^{(j)} \right\} - \mathbb{E}_W \left[ \mathbb{I} \left\{ A_{ir}^{(j)} \right\} \right] \right)^2 \right] \leq \mathbb{E}_W \left[ \mathbb{I} \left\{ A_{ir}^{(j)} \right\} \right] \leq \frac{2R}{\kappa \sqrt{2\pi}}.
\]

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

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

Therefore, with probability at least $1 - np \exp \left( -m^2 R \right)$ it holds for all $i \in [n]$ and $j \in [p]$ that:

\[
|P_{ij}^\perp| = \sum_{r=1}^{m} \mathbb{I} \left\{ A_{ir}^{(j)} \right\} \leq 3m^2 R.
\]

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 $w_{0,r} \sim \mathcal{N} \left( 0, \kappa^2 I \right)$ 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 $\kappa \leq 1$, $R \leq \kappa \sqrt{\frac{d}{2\pi}}$. 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 $R \leq \frac{2 \kappa^2}{\sqrt{2}}$. With probability at least $1 - ne^{-\frac{m}{2}}$ over initialization, it holds for all $i \in [n]$ that:

\[
\sum_{r=1}^{m} \langle w_{0,r}, x_i \rangle^2 \leq 2m^2 R^2
\]

\[
\sum_{i=1}^{n} \sum_{r=1}^{m} \langle w_{0,r}, x_i \rangle^2 \leq 2mn^2 R^2.
\]

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

\[
\mathbb{E}_{W_0, \hat{a}} \left[ \|y - u_0\|^2 \right] \leq (p^{-1} + C^2) n.
\]

**Proof:** It is obvious that $\mathbb{E}_{W_0, \hat{a}} \left[ u_0^{(i)} \right] = 0$ for all $i \in [n]$. Moreover:

\[
\mathbb{E}_{W_0, \hat{a}} \left[ u_0^{(i)} \right] = \sum_{r,r'=1}^{p} \sum_{j,j'=1}^{q} \mathbb{E}_{a_{rr'}, a_{jj'}} \left[ \mathbb{E}_W \left[ \sigma \left( \langle \hat{x}_i^{(j)}, w_{0,r} \rangle \right) \sigma \left( \langle \hat{x}_i^{(j')}, w_{0,r'} \rangle \right) \right] \right]
\]

\[
= \frac{1}{p^2m} \sum_{r=1}^{p} \sum_{j=1}^{q} \mathbb{E}_W \left[ \sigma \left( \langle \hat{x}_i^{(j)}, w_{0,r} \rangle \right) \right]^2
\]

\[
\leq \frac{1}{p^2m} \sum_{r=1}^{p} \sum_{j=1}^{q} \mathbb{E}_W \left[ \langle \hat{x}_i^{(j)}, w_{0,r} \rangle \right]
\]

\[
\leq p^{-1}
\]

Therefore:

\[
\mathbb{E}_{W_0, \hat{a}} \left[ \|y - u_0\|^2 \right] = \sum_{i=1}^{n} \mathbb{E}_{W_0, \hat{a}} \left[ u_0^{(i2)} \right] = \sum_{i=1}^{n} \left( \mathbb{E}_{W_0, \hat{a}} \left[ u_0^{(i2)} \right] + y_i^2 \right) \leq (p^{-1} + C^2) n
\]
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:

\[ H(t)_{ii'} = \sum_{r=1}^{m} \sum_{j,j'=1}^{p} a_{rj} a_{rj'} \langle \tilde{x}_i^{(j)}, \tilde{x}_{i'}^{(j')} \rangle \mathbb{I} \{ \tilde{x}_i^{(j)}; w_{r,t} \} \mathbb{I} \{ \tilde{x}_{i'}^{(j')}; w_{r,t} \}. \]

Moreover, let the infinite width NTK be defined as

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

Let \( \lambda_0 = \lambda_{\min}(H^{\infty}) \). Note that since \( \tilde{x}_i \parallel \tilde{x}_{i'} \) for \( i \neq i' \). Thus \( \tilde{x}_i^{(j)} \parallel \tilde{x}_{i'}^{(j)} \) for \( i \neq i' \). () shows that the matrix \( H(j) \), as defined below, is positive definite for all \( j \in [p] \):

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

Since \( H^{\infty} = p^{-2} \sum_{j=1}^{p} \tilde{H}(j)^{\infty} \), we have that \( H^{\infty} \) 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 \( \| w_{r,t} - w_{r,0} \|_2 \leq R := \Theta \left( \frac{\lambda_0}{m} \right) \). Then with probability at least \( 1 - \delta \) we have that for all \( t \):

\[ \lambda_{\min}(H(t)) \geq \frac{\lambda_0}{2}. \]

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

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

Moreover, we have that:

\[
\left| \sum_{j,j'=1}^{p} a_{rj} a_{rj'} \langle \tilde{x}_i^{(j)}, \tilde{x}_{i'}^{(j')} \rangle \mathbb{I} \{ \tilde{x}_i^{(j)}; w_{r,0} \} \mathbb{I} \{ \tilde{x}_{i'}^{(j)}; w_{r,0} \} \right| \leq 1.
\]

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

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

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

\[ \| H(0)_{i,i'} - H_{i,i'}^{\infty} \| \leq \log \frac{n}{\delta} \sqrt{m}, \]

which implies that:

\[ \| H(0) - H^{\infty} \| \leq \| H(0) - H^{\infty} \|_F \leq \frac{n \log \delta^{-1} + \log n}{\sqrt{m}}. \]

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

\[ \| H(t) - H^{\infty} \| \leq \frac{\lambda_0}{4}. \]
Now, we move on to bound $\|H(t) - H(0)\|$. We have that:

$$H(t)_{i,i'} - H(0)_{i,i'} = \sum_{r=1}^{m} \sum_{j,j'=1}^{p} a_{rj} a_{rj'} \langle x^{(j)}_{r}, x^{(j')}_{r} \rangle z^{(j,j')}_{r,i,i'},$$

with

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

We observe that $|z^{(j,j')}_{r,i,i'}|$ only if $A^{(j)}_{ir} \lor A^{(j')}_{ir}$. Therefore:

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

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

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

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

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

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

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

Moreover:

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

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

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

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

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

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

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

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

This implies that:

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

Thus, $\|H(t) - H(0)\|_2 \leq \frac{\lambda_{\min}}{4}$ as long as $R = O \left( \frac{\lambda_{\min}}{\pi} \right)$. This shows that $\lambda_{\min} (H(t)) \geq \frac{\lambda_{\min}}{2}$ 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

$$w_{r,t+1} = w_{r,t} - \eta g_{r,t} - \delta_s; \quad g_{r,t} = \sum_{i=1}^{n} \sum_{j=1}^{p} a_{rj} \left( \tilde{u}_{r,t}^{(i)} - \frac{y}{N_{r,t}} \right) \tilde{x}_i \parallel \tilde{x}_i^{(j)}; w_{r,t}$$

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

$$\tilde{u}_{r,t}^{(i)} = \frac{N_{r,t}}{N_{r,t}} \sum_{s=1}^{S} \sum_{t'=1}^{p} m_{r,t,u_{i t}} = \sum_{r'=1}^{m} \sum_{j=1}^{p} \left( \frac{N_{r,t}}{N_{r,t}} \sum_{s=1}^{S} m_{r,t,m_{r,t}} \right) a_{rj} \sigma \left( \langle \tilde{x}_i^{(j)}, w_{r,t} \rangle \right)$$

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

$$\mathbb{E}_M \left[ \tilde{u}_{r,t}^{(i)} \right] = \sum_{s=1}^{S} \sum_{t'=1}^{p} \sum_{j=1}^{p} \mathbb{E}_M \left[ \tilde{m}_{r,t,m_{r,t}}^{(i)} \left( \frac{N_{r,t}}{N_{r,t}} \right) a_{rj} \sigma \left( \langle \tilde{x}_i^{(j)}, w_{r,t} \rangle \right) \right]$$

$$= \xi \theta \sum_{r'=1}^{m} \sum_{j=1}^{p} a_{rj} \sigma \left( \langle \tilde{x}_i^{(j)}, w_{r,t} \rangle \right) + (1 - \xi) \theta \sum_{j=1}^{p} a_{rj} \sigma \left( \langle \tilde{x}_i^{(j)}, w_{r,t} \rangle \right)$$

$$= \theta u_{i t}^{(i)} + (1 - \xi) \theta \sum_{j=1}^{p} a_{rj} \sigma \left( \langle \tilde{x}_i^{(j)}, w_{r,t} \rangle \right)$$

Therefore,

$$\mathbb{E}_M \left[ g_{r,t} \right] = \sum_{i=1}^{n} \sum_{j=1}^{p} a_{rj} \mathbb{E}_M \left[ \tilde{u}_{r,t}^{(i)} - \frac{y}{N_{r,t}} \right] \tilde{x}_i \parallel \tilde{x}_i^{(j)}; w_{r,t}$$

$$= \xi^{-1} \theta \nabla_{w_r} L (W_t) + (1 - \xi) \theta \sum_{i=1}^{n} \sum_{j'=1}^{p} a_{rj} \epsilon_{r,t}^{(i)} \tilde{x}_i \parallel \tilde{x}_i^{(j)}; w_{r,t}$$

Now, we have

$$\left\| \epsilon_{r,t}^{(i)} \right\| \leq \frac{1}{\sqrt{m}} \left\| w_{r,t} \right\|_2; \quad \left\| \epsilon_{r,t} \right\|_2 \leq \frac{n}{\sqrt{m}} \left\| \epsilon_{r,t}^{(i)} \right\| \leq \frac{n}{m} \left\| w_{r,t} \right\|_2$$

Moreover, we would like to investigate the norm and norm squared of the gradient. In particular, we first notice that, under the case of $\frac{1}{\sqrt{m}} = 1$, we have

$$g_{r,t} = \xi^{-1} \nabla_{w_r} L (W_t) + \sum_{i=1}^{n} \sum_{j'=1}^{p} a_{rj} \left( \tilde{u}_{r,t}^{(i)} - u_{i t}^{(i)} \right) \tilde{x}_i \parallel \tilde{x}_i^{(j)}; w_{r,t}$$

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

$$\mathbb{E}_M \left[ u_{r,r',t} | N_{r,r'}^{+} = 1 \right] = \begin{cases} \xi & \text{if } r \neq r' \\ 1 & \text{if } r = r' \end{cases} \quad \text{Var} \left( u_{r,r',t} | N_{r,r'}^{+} = 1 \right) = \begin{cases} \frac{\theta - \xi^2}{s} & \text{if } r \neq r' \\ 0 & \text{if } r = r' \end{cases}$$
Therefore
\[
\mathbb{E}_M, \left[ \| \hat{u}_{r,t} - u_t \|_2^2 \mid N_{r,t}^+ = 1 \right] \\
= n \mathbb{E}_M, \left[ \left( \sum_{r'=1}^{m} \sum_{j=1}^{p} (\nu_{r,r',t} - \xi) a_{r,j} \sigma \left( \langle \hat{x}_i^{(j)}, w_{r',t} \rangle \right) \right)^2 \mid N_{r,t}^+ = 1 \right] \\
= \frac{1}{m} \sum_{i=1}^{n} \sum_{r=1}^{m} \mathbb{E}_M, \left[ (\nu_{r,r',t} - \xi)^2 \mid N_{r,t}^+ = 1 \right] \left( \sum_{j=1}^{p} a_{r,j} \sigma \left( \langle \hat{x}_i^{(j)}, w_{r',t} \rangle \right) \right)^2 \\
\leq \frac{1}{m} \sum_{i=1}^{n} \sum_{r=1}^{m} \operatorname{Var} \left[ \nu_{r,r',t} \mid N_{r,t}^+ = 1 \right] \sum_{j=1}^{p} \sigma \left( \langle \hat{x}_i^{(j)}, w_{r',t} \rangle \right)^2 \\
\leq \frac{1}{m} \sum_{i=1}^{n} \sum_{r=1}^{m} \sum_{r' \neq r} (\nu_{r,r',t}^2 \mid N_{r,t}^+ = 1) \sum_{j=1}^{p} \sigma \left( \langle \hat{x}_i^{(j)}, w_{r',t} \rangle \right)^2 \\
\leq \frac{1}{m} \sum_{i=1}^{n} \sum_{r=1}^{m} \sum_{r' \neq r} (\nu_{r,r',t}^2 \mid N_{r,t}^+ = 1) \sum_{j=1}^{p} \sigma \left( \langle \hat{x}_i^{(j)}, w_{r',t} \rangle \right)^2 \\
\leq \frac{1}{m} \sum_{i=1}^{n} \sum_{r=1}^{m} \sum_{r' \neq r} \left( \sum_{j=1}^{p} \left( \hat{x}_i^{(j)} - \frac{\theta - \xi^2}{mS} \| u_t - y \|_2 \right)^2 \right) \\
\sum_{r=1}^{m} \left( \sum_{j=1}^{p} \left( \hat{x}_i^{(j)} - \frac{\theta - \xi^2}{mS} \| u_t - y \|_2 \right)^2 \right) \leq 2mn\kappa^2 - mnpR^2.
\]
With high probability it holds that
\[
\sum_{r=1}^{m} \left( \sum_{j=1}^{p} \left( \hat{x}_i^{(j)} - \frac{\theta - \xi^2}{mS} \| u_t - y \|_2 \right)^2 \right) \leq 2mn\kappa^2 - mnpR^2.
\]
Thus, with sufficiently large \( m \), the second term is always smaller than the first term, and we have
\[
\mathbb{E}_M, \left[ \| \hat{u}_{r,t} - u_t \|_2^2 \mid N_{r,t}^+ = 1 \right] \leq 8n\kappa^2(\theta - \xi^2)S^{-1}
\]
Now, we can compute that
\[
\mathbb{E}_M, \left[ \| g_{r,t} \|_2 \mid N_{r,t}^+ = 1 \right] = \mathbb{E}_M, \left[ \left( \sum_{r=1}^{m} \sum_{j=1}^{p} a_{r,j} \left( \hat{u}_{r,t}^{(j)} - u_t^{(j)} \right) \right) \| \hat{x}_i^{(j)} - \langle \hat{x}_i^{(j)}, w_{r,t} \rangle \|_2 \mid N_{r,t} = 1 \right] + \\
\frac{\kappa^2 \theta}{m} \| \nabla_{w_i} \mathcal{L} (W_t) \|_2 \\
= \sqrt{\frac{n}{m}} \mathbb{E}_M, \left[ \| \hat{u}_{r,t} - u_t \|_2 \mid N_{r,t} = 1 \right] + \theta \sqrt{\frac{n}{m}} \| u_t - y \|_2 \\
\leq n\kappa \sqrt{\frac{\theta - \xi^2}{mS}} + \theta \sqrt{\frac{n}{m}} \| u_t - y \|_2
\]
And we know that \( g_{r,t} = 0 \) when \( N_{r,t}^+ = 0 \). Therefore,
\[
\mathbb{E}_M, \left[ \| g_{r,t} \|_2 \right] \leq \theta n\kappa \sqrt{\frac{\theta - \xi^2}{mS}} + \theta^2 \sqrt{\frac{n}{m}} \| u_t - y \|_2
\]
Similarly
\[
\mathbb{E}_M, \left[ \| g_{r,t} \|_2^2 \mid N_{r,t}^+ = 1 \right] = \frac{2n}{m} \mathbb{E}_M, \left[ \| \hat{u}_{r,t} - u_t \|_2^2 \mid N_{r,t} = 1 \right] + \frac{2\theta^2 n}{m} \| u_t - y \|_2^2 \\
\leq \frac{16n^2 \kappa^2 (\theta - \xi^2)}{mS} + \frac{2\theta^2 n}{m} \| u_t - y \|_2^2
\]
and thus
\[
\mathbb{E}_M, \left[ \| g_{r,t} \|_2^2 \right] \leq \frac{16\theta n^2 \kappa^2 (\theta - \xi^2)}{mS} + \frac{2\theta^3 n}{m} \| u_t - y \|_2^2
\]
4.5 Step-wise Convergence

Consider

\[
\mathbb{E}_M \left[ \| u_{t+1} - y \|^2 \right] = \| u_t - y \|^2 - 2 \langle u_t - u_{t+1}, u_t - y \rangle + \| u_t - u_{t+1} \|^2
\]

\[
= \| u_t - y \|^2 - 2 \langle I_{1,t}, u_t - y \rangle + \| u_t - u_{t+1} \|^2
\]

\[
\leq \| u_t - y \|^2 - 2 \langle I_{1,t}, u_t - y \rangle + 2 \| I_{2,t} \|_2 \| u_t - y \|_2 + \| u_t - u_{t+1} \|^2
\]

Let \( P_{ij} = \{ r \in [m] : -A_{ir}^{(j)} \} \). Here \( I_{1,t} \) and \( I_{2,t} \) are characterized as in previous work.

\[
I_{1,t}^{(i)} = \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left( \sigma \left( \langle \hat{x}_{i}^{(j)}, w_{r,t} \rangle \right) - \sigma \left( \langle \hat{x}_{i}^{(j)}, w_{r,t+1} \rangle \right) \right)
\]

\[
I_{2,t}^{(i)} = \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \left( \sigma \left( \langle \hat{x}_{i}^{(j)}, w_{r,t} \rangle \right) - \sigma \left( \langle \hat{x}_{i}^{(j)}, w_{r,t+1} \rangle \right) \right)
\]

We first bound the magnitude of \( I_{2,t} \)

\[
| I_{2,t}^{(i)} | = \frac{1}{p \sqrt{m}} \sum_{j=1}^{p} \sum_{r \in P_{ij}} \left( \langle \hat{x}_{i}^{(j)}, w_{r,t} \rangle - \langle \hat{x}_{i}^{(j)}, w_{r,t+1} \rangle \right) \left| \sigma \left( \langle \hat{x}_{i}^{(j)}, w_{r,t} \rangle \right) - \sigma \left( \langle \hat{x}_{i}^{(j)}, w_{r,t+1} \rangle \right) \right|
\]

\[
\leq \frac{1}{p \sqrt{m}} \sum_{j=1}^{p} \sum_{r \in P_{ij}} \| w_{r,t} - w_{r,t+1} \|_2
\]

\[
\leq \frac{\eta}{p \sqrt{m}} \sum_{j=1}^{p} \sum_{r \in P_{ij}} \| g_{r,t-\delta_t} \|_2
\]

\[
\leq \frac{\eta \sqrt{m}}{\sqrt{p}} \cdot 3m \kappa^{-1} R \cdot \left( n \kappa \sqrt{\frac{\theta - \xi^2}{mS}} + \sqrt{\frac{n}{m}} \| u_{t-\delta_t} - y \|_2 \right)
\]

\[
= 3 \kappa^{-1} \theta \eta \sqrt{n} R \| u_{t-\delta_t} - y \|_2 + 3 \eta R \sqrt{\frac{\theta - \xi^2}{S}}
\]

Thus

\[
\mathbb{E}_M \left[ \| I_{2,t} \|_2 \right] \leq \mathbb{E}_M \left[ \sqrt{n} \max_{i \in [n]} | I_{2,t}^{(i)} | \right]
\]

\[
\leq \frac{\eta \sqrt{n}}{\sqrt{m}} \max_{i \in [n]} \sum_{j=1}^{p} \mathbb{E}_M \left[ \| g_{r,t-\delta_t} \|_2 \right]
\]

\[
\leq \eta \sqrt{\frac{n}{m}} \cdot 3m \kappa^{-1} R \cdot \left( n \kappa \sqrt{\frac{\theta - \xi^2}{mS}} + \sqrt{\frac{n}{m}} \| u_{t-\delta_t} - y \|_2 \right)
\]

\[
= 3 \kappa^{-1} \theta \eta \sqrt{n} R \| u_{t-\delta_t} - y \|_2 + 3 \theta \eta R \sqrt{\frac{\theta - \xi^2}{S}}
\]

Therefore, given assumption \( \| u_{t-\delta_t} - y \|_2 \geq \| u_t - y \|_2 \)

\[
\mathbb{E}_M \left[ \| I_{2,t} \|_2 \| u_t - y \|_2 \right] \leq 6 \theta \eta \kappa^{-1} R \| u_{t-\delta_t} - y \|_2^2 + 3 S^{-1} \theta (\theta - \xi^2) \eta \kappa^2 R
\]

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

\[
\mathbb{E}_M \left[ \| I_{2,t} \|_2 \| u_t - y \|_2 \right] = O (\theta \eta \lambda_0) \| u_{t-\delta_t} - y \|_2^2 + O (\theta \eta \lambda_0 (\theta - \xi^2) S^{-1} \kappa^2)\)
As in previous work, $I^{(i)}_{1,t}$ can be written as

\[
\mathbb{E}_{\mathcal{M}_t} \left[ I^{(i)}_{1,t} \right] = \xi \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \mathbb{E}_{\mathcal{M}_t} \left[ \sigma \left( \langle \hat{x}_i^{(j)}, w_{r,t} \rangle \right) - \sigma \left( \langle \hat{x}_i^{(j)}, w_{r,t+1} \rangle \right) \right] \\
= \xi \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_i^{(j)}, \mathbb{E}_{\mathcal{M}_t} [w_{r,t} - w_{r,t+1}] \rangle \mathbb{P} \left\{ \hat{x}_i^{(j)} : w_{r,t} \right\} \\
= \xi \eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_i^{(j)}, \mathbb{E}_{\mathcal{M}_t} [g_{r,t} - g_{r,t}] \rangle \mathbb{P} \left\{ \hat{x}_i^{(j)} : w_{r,t} \right\} \\
= \xi \eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_i^{(j)}, \mathbb{E}_{\mathcal{M}_t} [g_{r,t} - g_{r,t}] \rangle \mathbb{P} \left\{ \hat{x}_i^{(j)} : w_{r,t} \right\} + \\
= \xi \eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_i^{(j)}, \mathbb{E}_{\mathcal{M}_t} [g_{r,t} - g_{r,t}] \rangle \mathbb{P} \left\{ \hat{x}_i^{(j)} : w_{r,t} \right\} \\
= \eta \theta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_i^{(j)}, \nabla_w \mathcal{L} (W_t) \rangle \mathbb{P} \left\{ \hat{x}_i^{(j)} : w_{r,t} \right\} \\
+ \xi (1 - \xi) \theta \eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_i^{(j)}, \epsilon_{r,t} \rangle \mathbb{P} \left\{ \hat{x}_i^{(j)} : w_{r,t} \right\} \\
+ \eta \theta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_i^{(j)}, \nabla_w \mathcal{L} (W_{t-\delta_t}) - \nabla_w \mathcal{L} (W_t) \rangle \mathbb{P} \left\{ \hat{x}_i^{(j)} : w_{r,t} \right\} \\
+ \xi (1 - \xi) \theta \eta \sum_{j=1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_i^{(j)}, \epsilon_{r,t-\delta_t} - \epsilon_{r,t} \rangle \mathbb{P} \left\{ \hat{x}_i^{(j)} : w_{r,t} \right\}
\begin{align*}
\eta \theta \xi \sum_{i' = 1}^{n} \sum_{j = 1}^{p} \sum_{r \in P_{ij}} (u_{t}^{i'} - y_{i'}) a_{rj} \alpha_{rj'} \langle \hat{x}_{i}^{(j)}; \hat{x}_{i'}^{(j')} \rangle \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i'}^{(j')} \} & : \mathbb{I} \{ \hat{x}_{i'}^{(j')}; \hat{x}_{i}^{(j)} \} \\
+ \xi (1 - \xi) \eta \sum_{j = 1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_{i}^{(j)}; \epsilon_{r,t} \rangle & \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i'}^{(j')} \} \\
+ \eta \theta \sum_{j = 1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_{i}^{(j)}; \nabla w_{t} \mathcal{L} (W_{t-\delta t}) - \nabla w_{t} \mathcal{L} (W_{t}) \rangle & \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i'}^{(j')} \} \\
+ \xi (1 - \xi) \eta \sum_{j = 1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_{i}^{(j)}; \epsilon_{r,t-\delta t} - \epsilon_{r,t} \rangle & \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i'}^{(j')} \} \\
= \eta \theta \sum_{i' = 1}^{n} \left( H(t)_{i'i'} - H(t)_{ii'} \right) (u_{t}^{i'} - y_{i'}) \\
+ \xi (1 - \xi) \eta \sum_{j = 1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_{i}^{(j)}; \epsilon_{r,t} \rangle & \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i'}^{(j')} \} \\
+ \eta \theta \sum_{j = 1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_{i}^{(j)}; \nabla w_{t} \mathcal{L} (W_{t-\delta t}) - \nabla w_{t} \mathcal{L} (W_{t}) \rangle & \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i'}^{(j')} \} \\
+ \xi (1 - \xi) \eta \sum_{j = 1}^{p} \sum_{r \in P_{ij}} a_{rj} \langle \hat{x}_{i}^{(j)}; \epsilon_{r,t-\delta t} - \epsilon_{r,t} \rangle & \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i'}^{(j')} \} \\
\end{align*}

Note that:

\begin{align*}
|\gamma_{1,i,t}| & \leq (1 - \xi) \eta m^{-\frac{1}{2}} \sum_{r = 1}^{m} \| \epsilon_{r,t} \| \\
& \leq (1 - \xi) \eta m^{-\frac{1}{2}} \| W_{t} \|_{F} \\
& \leq O \left( (1 - \xi) \eta m k \sqrt{\frac{d}{m}} \right)
\end{align*}

Given that:

\begin{align*}
\| (u_{t-\delta t}^{i} - u_{t}^{i}) \|_{2} & = \left\| \xi \sum_{r = 1}^{m} \sum_{j = 1}^{p} a_{rj} \sigma \left( \langle \hat{x}_{i}^{(j)}; \hat{x}_{i,t-\delta t} \rangle \right) - \xi \sum_{r = 1}^{m} \sum_{j = 1}^{p} a_{rj} \sigma \left( \langle \hat{x}_{i}^{(j)}; \hat{x}_{i,t} \rangle \right) \right\|_{2} \\
& \leq \xi \sum_{r = 1}^{m} \sum_{j = 1}^{p} \| a_{rj} \hat{x}_{i}^{(j)} \|_{2} \| \hat{x}_{i,t-\delta t} - \hat{x}_{i,t} \|_{2} \leq \frac{\xi}{\sqrt{m}} R_{E}
\end{align*}

We show:

\begin{align*}
\| \nabla w_{t} \mathcal{L} (W_{t-\delta t}) - \nabla w_{t} \mathcal{L} (W_{t}) \|_{2} & = \left\| \xi \sum_{i = 1}^{n} \sum_{j = 1}^{p} (u_{t-\delta t}^{i} - y_{i}) a_{rj} \hat{x}_{i}^{(j)} \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i,t-\delta t} \} - \xi \sum_{i = 1}^{n} \sum_{j = 1}^{p} (u_{t}^{i} - y_{i}) a_{rj} \hat{x}_{i}^{(j)} \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i,t} \} \right\|_{2} \\
& \leq \xi \sum_{i = 1}^{n} \sum_{j = 1}^{p} \| a_{rj} \hat{x}_{i}^{(j)} \|_{2} \| (u_{t-\delta t}^{i} - y_{i}) \| \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i,t-\delta t} \} - \| (u_{t}^{i} - y_{i}) \| \mathbb{I} \{ \hat{x}_{i}^{(j)}; \hat{x}_{i,t} \} \|_{2} \\
& \leq \xi \sum_{i = 1}^{n} \sum_{j = 1}^{p} \| a_{rj} \hat{x}_{i}^{(j)} \|_{2} \left( \| (u_{t-\delta t}^{i} - u_{t}^{i}) \| + \| (u_{t}^{i} - y_{i}) \| \right) \\
& \leq \xi^{2} \frac{n}{m} R_{E} + \xi \sqrt{\frac{n}{m}} \| u_{t} - y \|_{2}
\end{align*}
Thus,

\[ \| \gamma_{2,i,t} \|_2 \leq 2\eta \theta n m^{-2} \xi^2 R_E + \eta \theta \xi n m^{-\frac{1}{2}} \| u_t - y \|_2 \]

Given that:

\[
\left\| \tilde{e}_{r,t}^{(i)} - \tilde{e}_{r,t}^{(j)} \right\|_2 = \left\| \sum_{j=1}^{p} a_{r,j} \sigma \left( (\tilde{X}_i^{(j)} ; w_{r,t-\delta_t}) \right) - \sum_{j=1}^{p} a_{r,j} \sigma \left( (\tilde{X}_i^{(j)} ; w_{r,t}) \right) \right\|_2 \\
\leq \sum_{j=1}^{p} \left\| a_{r,j} \tilde{X}_i^{(j)} \right\|_2 \left( \| w_{r,t-\delta_t} - w_{r,t} \|_2 + \| w_{r,t} \|_2 \right) \\
\leq \kappa \sqrt{\frac{2d}{m}} + \frac{1}{\sqrt{m}} R_E
\]

We show:

\[
\left\| \tilde{e}_{r,t}^{(i)} - \tilde{e}_{r,t}^{(j)} \right\|_2 = \left\| \sum_{i=1}^{n} \sum_{j=1}^{p} a_{r,j} \tilde{e}_{r,t}^{(i)} \tilde{X}_i^{(j)} \| \cdot \left\{ \tilde{X}_i^{(j)} ; w_{r,t-\delta_t} \right\} - \sum_{i=1}^{n} \sum_{j=1}^{p} a_{r,j} \tilde{e}_{r,t}^{(i)} \tilde{X}_i^{(j)} \| \cdot \left\{ \tilde{X}_i^{(j)} ; w_{r,t} \right\} \right\|_2 \\
\leq \sum_{i=1}^{n} \sum_{j=1}^{p} \left\| a_{r,j} \tilde{X}_i^{(j)} \right\|_2 \left( \left\| \left( \tilde{e}_{r,t}^{(i)} - \tilde{e}_{r,t}^{(j)} \right) \|_2 \right) + \left\| \left( \tilde{e}_{r,t}^{(i)} \right) \|_2 \right) \\
\leq \frac{n}{m} \left( \kappa \sqrt{\frac{2d}{m}} + \frac{1}{\sqrt{m}} R_E + \frac{1}{\sqrt{m}} \kappa \sqrt{2d} \right)
\]

Thus,

\[ \| \gamma_{3,i,t} \|_2 \leq \sqrt{2}\xi (1 - \xi) \theta n m^{-3} \kappa \sqrt{d} + 2\xi (1 - \xi) \theta n m^{-3} R_E + \sqrt{2} \xi (1 - \xi) \theta n m^{-1} \kappa \sqrt{d} \]

This implies that:

\[
\mathbb{E}_{M_t} [ (I_{1,t} ; u_t - y) ] = \eta \theta \sum_{i',t'=1}^{n} \left( u_{i'}^{(i)} - y_i \right) \left( (H(t))_{i' t'} - (H(t))_{i' t'}^{\perp} \right) \left( u_{i'}^{(i)} - y_i \right) \\
+ \sum_{i=1}^{n} \gamma_{1,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) + \sum_{i=1}^{n} \gamma_{2,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) + \sum_{i=1}^{n} \gamma_{3,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) \\
= \eta \theta \left( u_t - y \right) \left( (H(t)) - (H(t))^{\perp} \right) \left( u_t - y \right) \\
+ \sum_{i=1}^{n} \gamma_{1,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) + \sum_{i=1}^{n} \gamma_{2,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) + \sum_{i=1}^{n} \gamma_{3,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) \\
\geq \eta \theta \left( \lambda_{\min} (H(t)) - \lambda_{\max} (H(t)) \right) \left( u_t - y \right) \left( u_t - y \right) \\
- \sum_{i=1}^{n} \gamma_{1,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) - \sum_{i=1}^{n} \gamma_{2,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) - \sum_{i=1}^{n} \gamma_{3,i,t} \left( u_{i}^{(i)} - y_i \right) \left( u_{i}^{(i)} - y_i \right) \\
\geq \eta \theta \left( \lambda_{\min} (H(t)) - \lambda_{\max} (H(t)) \right) \left( u_t - y \right) \left( u_t - y \right) \\
- \sqrt{n} \max_{\delta} \| \gamma_{1,i,t} \| \left( u_t - y \right) \left( u_t - y \right) \\
- 2\eta \theta - \frac{n^2}{3} m^{-3} \xi^2 R_E \left( u_t - y \right) \left( u_t - y \right) - \eta \theta \xi n m^{-\frac{1}{2}} \left( u_t - y \right) \left( u_t - y \right) \\
- \sqrt{2} \xi (1 - \xi) \theta n m^{-1} \kappa \sqrt{d} \left( u_t - y \right) \left( u_t - y \right) - 2\xi (1 - \xi) \theta n \xi m^{-\frac{1}{2}} R_E \left( u_t - y \right) \left( u_t - y \right) \\
\geq \eta \theta \left( \lambda_{\min} (H(t)) - \lambda_{\max} (H(t)) \right) \left( u_t - y \right) \left( u_t - y \right) \\
- O \left( \frac{\xi^2 (1 - \xi)^2 \theta n^3 \kappa d}{m \lambda_0} + \frac{\eta \theta^2 n^3 \xi^2 R_E}{m^4 \lambda_0} \right) + \frac{\xi^2 (1 - \xi)^2 \theta^2 n^2 \kappa^2 d}{m^3 \lambda_0} +
\[
\frac{\xi^2(1-\xi)2\theta^2\eta^2n^2R^2_{E}}{m^3\lambda_0} + \frac{\xi^2(1-\xi)2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0}
\]

For \( H(t)^{\perp} \) we have that
\[
\lambda_{\text{max}} \left( H(t)^{\perp} \right)^2 \leq \|H(t)^{\perp}\|_F^2
\]
\[
\leq \frac{n^2}{m^2} \left( \max_{ij} |P_{ij}| \right)^2
\]
\[
\leq n^2\kappa^{-2}R^2
\]
Choosing \( R = O \left( \frac{\kappa \lambda_0}{\theta} \right) \) gives
\[
\lambda_{\text{max}} \left( H(t)^{\perp} \right) \leq O \left( \lambda_0 \right)
\]
We require \( m \geq \Omega \left( \frac{\lambda_0}{\kappa} \right) \), such that
\[
m\theta^{-\frac{3}{2}} \leq O \left( \lambda_0 \right)
\]
Thus, Plugging in \( \lambda_{\text{min}} \left( H(t) \right) \geq \frac{\lambda_0}{2} \), we have
\[
\mathbb{E}_{M_t} \left[ \|I_{1,t}, u_t - y\| \right] \geq \eta\theta\lambda_0 \left( \frac{1}{2} - O \left( 1 \right) \right) \|u_t - y\|_2^2
\]
\[
- O \left( \frac{\xi^2(1-\xi)2\theta^2n^3\kappa^2d}{m^3\lambda_0} + \frac{\eta^22\theta^2n\kappa\lambda_0\xi^2E^2}{m^3} + \frac{\xi^2(1-\xi)2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0} + \frac{\xi^2(1-\xi)2\theta^2\eta^2n^2\kappa^2d}{m^3\lambda_0} \right).
\]
Lastly, we analyze the last term in the quadratic expansion
\[
\mathbb{E}_{M_t} \left[ \|u_t - u_{t+1}\|_2^2 \right] = \sum_{i=1}^{n} \mathbb{E}_{M_t} \left[ \left( u^{(i)}_t - u^{(i)}_{t+1} \right)^2 \right]
\]
\[
\leq p^{-1} \sum_{i=1}^{n} \sum_{j=1}^{p} \sum_{r=1}^{m} \mathbb{E}_{M_t} \left[ \left( \sigma \left( \langle \hat{x}^{(j)}, w_{r,t} \rangle \right) - \sigma \left( \langle \hat{x}^{(j)}, w_{r,t+1} \rangle \right) \right)^2 \right]
\]
\[
\leq p^{-1} \eta^2 \sum_{i=1}^{n} \sum_{j=1}^{p} \sum_{r=1}^{m} \mathbb{E}_{M_t} \left[ \|g_{r,t-\delta_t}\|_2^2 \right]
\]
\[
\leq O \left( \theta^3\eta^2n^2 \right) \|u_{t-\delta_t} - y\|_2^2 + O \left( \theta\eta^2n^2\kappa^2 \left( \theta - \xi^2 \right) S^{-1} \right)
\]
Letting \( \eta = O \left( \frac{\lambda_0}{m\theta} \right) \) gives
\[
\mathbb{E}_{M_t} \left[ \|u_t - u_{t+1}\|_2^2 \right] \leq O \left( \theta\eta\lambda_0 \right) \|u_{t-\delta_t} - y\|_2^2 + O \left( \theta\eta\lambda_0\kappa^2 \left( \theta - \xi^2 \right) S^{-1} \right)
\]
Putting all three terms together and as \( \|u_{t-\delta_t} - y\|_2^2 \leq \|u_{t-E} - y\|_2^2 \), we have that
\[
\mathbb{E}_{\mathfrak{M}_t} \left[ \| \mathbf{u}_{t+1} - \mathbf{y} \|_2^2 \right] \leq \| \mathbf{u}_t - \mathbf{y} \|_2^2 - \eta \theta \lambda_0 (1 - O(1)) \| \mathbf{u}_t - \mathbf{y} \|_2^2 + O(\theta \eta \lambda_0) \| \mathbf{u}_{t-E} - \mathbf{y} \|_2^2 \\
+ O \left( \frac{\xi^2(1 - \xi)\theta \eta n^3 \kappa^2 d}{m\lambda_0} + \frac{\eta^2 \theta^2 n\kappa^2 \lambda_0 \xi^4 E^2}{m^4} + \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^3 \lambda_0} \right) \\
+ \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0} + \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0} + \frac{\eta \theta^2 \eta n^2 \kappa^2 \lambda_0}{S} (\theta - \xi^2) \right) \\
\leq \| \mathbf{u}_t - \mathbf{y} \|_2^2 - \eta \theta \lambda_0 (1 - O(1)) \| \mathbf{u}_t - \mathbf{y} \|_2^2 + O(\theta \eta \lambda_0) (2 \| \mathbf{u}_{t-E} - \mathbf{u}_t \|_2^2 + 2 \| \mathbf{u}_t - \mathbf{y} \|_2^2) \\
+ O \left( \frac{\xi^2(1 - \xi)\theta \eta n^3 \kappa^2 d}{m\lambda_0} + \frac{\eta^2 \theta^2 n\kappa^2 \lambda_0 \xi^4 E^2}{m^4} + \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^3 \lambda_0} \right) \\
+ \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0} + \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0} + \frac{\eta \theta^2 \eta n^2 \kappa^2 \lambda_0}{S} (\theta - \xi^2) \right) \\
\leq \| \mathbf{u}_t - \mathbf{y} \|_2^2 - \eta \theta \lambda_0 (1 - O(1)) \| \mathbf{u}_t - \mathbf{y} \|_2^2 + O(\theta \eta \lambda_0) (8 \xi^2 R_k^2 + 2 \| \mathbf{u}_t - \mathbf{y} \|_2^2) \\
+ O \left( \frac{\xi^2(1 - \xi)\theta \eta n^3 \kappa^2 d}{m\lambda_0} + \frac{\eta^2 \theta^2 n\kappa^2 \lambda_0 \xi^4 E^2}{m^4} + \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^3 \lambda_0} \right) \\
+ \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0} + \frac{\xi^2(1 - \xi)\theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0} + \frac{\eta \theta^2 \eta n^2 \kappa^2 \lambda_0}{S} (\theta - \xi^2) \right)
\]

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

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

Thus,

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

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

$$E[M_t - 1][||w_{r,t} - w_{r,0}\|_2] \leq \sum_{t'=0}^{t-1} E[M_{t'}][||w_{r,t'+1} - w_{r,t'}\|_2]$$

$$\leq \eta \sum_{t'=0}^{t-1} E[M_{t'}][||g_{r,t'-\delta_t}||_2]$$

$$\leq \eta \sum_{t'=0}^{t-1} \left( \theta^2 \frac{n}{m} E[M_{t'-1}][||u_{t'-\delta_t} - y||_2] + \theta n \kappa \sqrt{\frac{\theta - \xi^2}{mS}} \right)$$

$$\leq \eta \theta^2 \frac{n}{m} \sum_{t'=0}^{t-1} E[M_{t'-1}][||u_{t'-\delta_t} - y||_2] + \eta \theta n \kappa \sqrt{\frac{\theta - \xi^2}{mS}}$$

$$\leq \eta \theta^2 \frac{n}{m} \frac{1}{\eta \theta \lambda_0} ||u_0 - y||_2$$

$$\leq \eta \theta^2 \frac{n}{m} \frac{1}{\eta \theta \lambda_0} ||u_0 - y||_2 + \eta \theta T \cdot O \left( \frac{\theta \eta \lambda_0 \xi^2 \lambda^2}{n^2} + \frac{\xi^2 (1 - \xi)^2 \theta \eta \lambda_0 \lambda^2}{m^4} + \frac{\eta^2 \theta^2 \kappa^2 \lambda_0 \xi^4}{m^4} + \frac{\xi^2 (1 - \xi)^2 \theta^2 \eta^2 n^2 \kappa^2 d}{m^3 \lambda_0} + \frac{\xi^2 (1 - \xi)^2 \theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0} \right)$$

$$\leq O \left( \eta \lambda_0 \lambda_0 \right) \left( \frac{n}{m} \right) ||u_0 - y||_2$$

$$\leq O \left( \eta \lambda_0 \lambda_0 \right) \left( \frac{n}{m} \right) ||u_0 - y||_2 + \eta \theta T \cdot O \left( \frac{\theta \eta \lambda_0 \xi^2 \lambda^2}{n^2} + \frac{\xi^2 (1 - \xi)^2 \theta \eta \lambda_0 \lambda^2}{m^4} + \frac{\eta^2 \theta^2 \kappa^2 \lambda_0 \xi^4}{m^4} + \frac{\xi^2 (1 - \xi)^2 \theta^2 \eta^2 n^2 \kappa^2 d}{m^3 \lambda_0} + \frac{\xi^2 (1 - \xi)^2 \theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0} \right)$$

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

$$E[\tilde{W}_{n,m,\lambda}][||w_{r,t} - w_{r,0}\|_2] \leq O \left( \eta \lambda_0 \lambda_0 \right) \frac{nm \kappa}{2}$$
With probability $1 - \delta$, it holds for all $t \in [T]$ that
\[
\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2 \leq O \left( \frac{t \lambda_0^2 K}{\delta n \sqrt{m}} \right) + \eta T \cdot O \left( \frac{\theta n \lambda_0^3 \xi \kappa^2}{n^2} + \frac{\xi^2 (1 - \xi)^2 \theta \eta m^3 \kappa^2 d}{m \lambda_0} + \frac{\eta^2 \theta^2 n \kappa \lambda_0 \kappa^4}{m^4} + \frac{\xi^2 (1 - \xi)^2 \theta^2 \eta^2 n^2 \kappa^2 d}{m^3 \lambda_0} \right) + \frac{\xi^2 (1 - \xi)^2 \theta^2 \eta^2 n^2 \kappa^2 d}{m^2 \lambda_0}.
\]

To enforce $\|\mathbf{w}_{r,t} - \mathbf{w}_{r,0}\|_2 \leq R := O \left( \frac{\kappa \lambda_0}{n} \right)$, we then require
\[
m = \Omega \left( \frac{n^3 K^2}{\lambda_0^2 \theta^2 \kappa^2 \max\{n, d\}} \right)
\]

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


