FedNLR: Federated Learning with Neuron-wise Learning Rates

Haozhao Wang
School of Computer Science and Technology, Huazhong University of Science and Technology
Wuhan, China

Peirong Zheng*
Department of Computing, The Hong Kong Polytechnic University
Hongkong, China

Xingshuo Han
Nanyang Technological University
Singapore, Singapore

Wenchao Xu
Department of Computing, The Hong Kong Polytechnic University
Hongkong, China

Ruixuan Li†
School of Computer Science and Technology, Huazhong University of Science and Technology
Wuhan, China

Tianwei Zhang
Nanyang Technological University
Singapore, Singapore

Abstract

Federated Learning (FL) suffers from severe performance degradation due to the data heterogeneity among clients. Some existing work suggests that the fundamental reason is that data heterogeneity can cause *local model drift*, and therefore proposes to calibrate the direction of local updates to solve this problem. Though effective, existing methods generally take the model as a whole, which lacks a deep understanding of how the neurons within deep classification models evolve during local training to form model drift. In this paper, we bridge this gap by performing an intuitive and theoretical analysis of the activation changes of each neuron during local training. Our analysis shows that the high activation of some neurons on the samples of a certain class will be reduced during local training when these samples are not included in the client, which we call *neuron drift*, thus leading to the performance reduction of this class. Motivated by this, we propose a novel and simple algorithm called FedNLR, which utilizes Neuron-wise Learning Rates during the FL local training process. The principle behind this is to enhance the learning of neurons bound to local classes on local data knowledge while reducing the decay of non-local classes knowledge stored in neurons. Experimental results demonstrate that FedNLR achieves state-of-the-art performance on federated learning with popular deep neural networks.

CCS Concepts

• **Computer systems organization → Distributed architectures**.

Keywords

Federated Learning; Neuron-wise Learning Rates; NonIID

*Haozhao Wang and Peirong Zheng contributed equally to this research.
†Corresponding author.

ACM Reference Format:
Haozhao Wang, Peirong Zheng, Xingshuo Han, Wenchao Xu, Ruixuan Li, and Tianwei Zhang. 2024. FedNLR: Federated Learning with Neuron-wise Learning Rates. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3672042

1 Introduction

Federated learning (FL) is emerging as a prominent framework to train deep neural networks (DNNs) via the collaboration among clients without sharing their original dataset [25, 32, 38, 42], and has been widely adopted in various applications such as medical image processing [9, 27, 44] and recommendation [2, 35]. The basic steps of FL [32] is to iteratively run the local training of the models in multiple clients separately and the global aggregation of all updated models in the server. Although the process of FL is easy to implement, the data among clients is usually statistically heterogeneous (i.e., not independently and identically distributed, NonIID), greatly degrading the performance.

Many works have been proposed to solve the NonIID problem. One of the representative categories is to explore the essence of performance degradation from the perspective of optimization, which considers the client drift (i.e., *model drift*) caused by the NonIID as the main reason for performance degradation. More specifically, Karimireddy et al. [17] claim that the optimum of multiple clients are different from each other and are also far away from the global optimum due to the NonIID data, thus resulting in the drift of optimization direction. Considering this, a series of approaches seek to achieve consistency of the local models across clients. For example, some methods add regularization on the local loss function to facilitate the local model to approach the global optimal model [1, 23]. Other works consider that the local gradient is biased and then correct the local gradient to align the global gradient [7, 17, 37] There are also some works that seek to calibrate the output of the last layer or the last-second layer to align them with the global model to achieve consistency [19, 22, 47]. Although these methods are effective, they generally view the deep neural network (DNN) as a whole, which lacks a deep understanding of how the neurons within the DNN evolve to form the issue of model drift.

In this paper, we seek to tackle the NonIID challenge of FL from the microscopic level by decomposing the DNN as a set of neurons to fully unleash FL performance potential. Specifically, we find that the set of high-activation neurons within a well-converged global DNN over each class is distinct from other classes, where these neurons we call *bound neurons* of this class. However, during the local training process, each client tends to overuse the DNN neurons by homogenizing as many neurons as possible to obtain high activation over its local data. Due to the data heterogeneity among clients, this may lead to the activation reduction of bound neurons of those data classes that are not included in this client, which we call *neuron drift*. As a consequence, how to prevent the activation of bound neurons over classes that are not included in the client from being reduced is the key to the global model learning the global data distribution.

Motivated by this discovery, we propose an effective yet simple method named FedNLR, which allows each client to locally train the deep neural network with neuron-wise learning rates. More specifically, FedNLR sets a specific learning rate for the parameters of each neuron based on their activation over the local data. The learning rate of a neuron is set to be smaller if its activation over the local data is smaller. The principle behind this is to enhance the learning of neurons bound to local classes on local data knowledge while restricting the decay of non-local classes knowledge stored in neurons. We conduct extensive experiments on various DNNs and datasets of which the experimental results show that our method outperforms existing methods. Besides, as far as we know, *our method is the first to solve the NonIID challenge from the perspective of adapting the local learning rate, which is orthogonal to existing approaches.* Our experimental results show that the performance of existing methods receives further improvement as combined with our method. Our contributions are:

- We are the first to investigate how the FL data heterogeneity affects the trained DNNs by decomposing the DNN into a set of neurons instead of viewing it as a whole. We identify that the activation value of a neuron on some certain class will be reduced during the local training process of a client if this client has no access to the samples of this class, resulting in performance degradation.
- We propose a novel method that allows each client to restrict the update of neurons bound to the other clients and enhances the learning of neurons bound to its local data during the local training process via leveraging neuron-wise learning rates. The learning rate for each neuron is adaptively set based on its activation over the local data.
- We provide a theoretical analysis that guarantees the convergence of the proposed method. Besides, the experiments conducted on various configurations such as different learning rates and different numbers of clients consistently demonstrate the effectiveness of our method.

2 Related Work

To train a global model, many previous efforts have been proposed to solve the NonIID challenge in FL [12, 14, 39]. There are mainly two types of work which include either adding a regularization item to the local loss or calibrating the local model update with the global information to achieve consistency across clients.

Regularization-based Consistency Some methods propose adding a regularization on the local loss function to achieve the consistency of local updated models [1, 23]. FedProx [23] penalizes the proximal term on the local objective to force the local update towards both the local optimum and the last global model. [1] proposed a dynamic regularizer that is based on the current local model and the received global model to achieve the same stationary point across all clients. [8] combines the local training process with the primal-dual algorithm to enhance the consistency among different variable models. FedSpeed [37] considers that the prox-term will also introduce the bias and then applies the prox-correction term on the current local updates to efficiently reduce the bias. [40] show that the hyper-parameters in the local update process also have an impact on the consistency and then regularize the local update. By encompassing the proximal term, [3] further propose a surrogate loss for the quadratic models and show that the local learning rate decay can balance the trade-off between the convergence rate and the inconsistency. [31] applies a local fixed-point to implicitly control the convergence of the local model. Based on the inconsistency of local update, [16] propose adaptively tuning the global step size via computing a regularized term of all local updates.

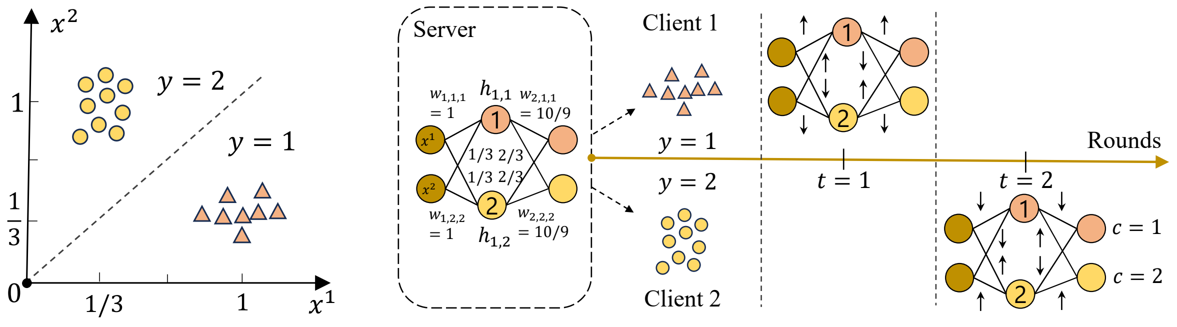
Calibration-based Consistency Other works consider that the local gradient is biased and then correct the local gradient to align the global gradient [13, 17, 37, 43]. [17] firstly demonstrates the drift of local update in FL and then propose Scaffold that exploits the bias of the local gradient to the global gradient to mitigate these drifts. [7] consider the drift between the local optimal model and the global optimal model essentially exists and propose learning the drift term to compensate local gradient. [29] propose incorporating the information of global gradient in the local training process such that the local bias can be mitigated. Some studies also introduce the momentum to the FL where the momentum also contains the information of past gradients from other clients such that the drift can be implicitly calibrated [18, 36, 41, 45]. Considering the last layer cares most about the classification, FedRS [26] utilizes an asymmetric loss function to calibrate the bias of the last-layer parameters of different classes. Similarly, there are also some works that seek to calibrate the output of the last layer or the last-second layer to align them with the global model to achieve consistency [19, 22, 47].

This paper focuses on the drift problem of local models across clients in federated learning. Different from previous works that usually view the training DNN as a whole and consider the drift of the whole local update, we aim to solve the neuron drift problem which is a fine-grained phenomenon of the client drift. To tackle this challenge, we adopt neuron-wise learning rates during the local training process, which is orthogonal to existing works.

3 Problem Formulation and Preliminaries

Problem Formulation. Our goal is to collectively train a global model in Federated Learning (FL) for a total of K clients. Each client k has exclusive access to their private local dataset $\mathbb{D}_k$ which consists of D_k data samples (x_i^n, y_i), where x_i represents the i-th input data sample and y_i belongs to a set of possible labels $\{1, 2, \dots, C\}$ with C denoting the total number of classes. The primary objective is to train a global model $\mathbf{w}$ that minimizes the total empirical loss across all local datasets. The loss function is formulated as:

$$\min_{\mathbf{w}} F(\mathbf{w}) := \sum_{k=1}^{K} \frac{D_k}{D} F_k(\mathbf{w}), \text{ where } F_k(\mathbf{w}) = \frac{1}{D_k} \sum_{i=1}^{D_k} f_k(\mathbf{w}; x_i, y_i), \quad (1)$$



(a) Data Distribution **(b) Training Process**

Figure 1: An example of two-client federated learning. One client is sampled in each round and each client only contains samples of one single label, i.e., client 1 with class 1 and client 2 with class 2. The data of classes 1 and 2 are denoted by triangle and circle respectively. The upward arrow represents that the value of the parameter is improved and the downward arrow has the opposite meaning. As can be seen, after training the model in client 1, the parameters corresponding to the bonded neurons of class 2 may be improperly tuned, leading to their reduced activation magnitude over the data of class 2. A similar phenomenon also exists in the client 2, which we call *neuron drift*.

where $D = \sum_{k=1}^{K} D_k$ denotes the total size of all local datasets, $F_k(\mathbf{w})$ represents the local loss for the k-th client, and $f_k(\cdot)$ usually adopts the cross-entropy loss which measures the discrepancy between the model's predictions and the actual ground truth labels.

Basics of deep neural network. We consider a deep neural network comprising L layers, where each layer l consists of M_l neurons. The model's weight parameters are represented as $\mathbf{w}$ and the parameters of the l-th layer are denoted by $\mathbf{w}_l$. For each neuron indexed as i in the l-th layer, we compute its activation output as $h_{l,i} = \sigma(\mathbf{w}_{l,i}^T \mathbf{h}_{l-1})$. In this expression, we use the commonly used activation function ReLU to signify $\sigma(\cdot)$, while $\mathbf{w}_{l,i}$ corresponds to the parameters specific to this neuron. Furthermore, $\mathbf{h_{l-1}}$ denotes the outputs of all neurons in the preceding $l-1$-th layer, characterized as $\mathbf{h_{l-1}} = [h_{l-1,1}, \ldots, h_{l-1,m_{l-1}}]$.

4 Neuron Drift

To make predictions, deep neural networks activate different neurons for the data of different classes. In particular, given a well-converged model, each class is bound to a specific subset of neurons, where the activation magnitude of these neurons over this class is larger than other classes. Formally, these bound neurons are defined as follows.

Definition 1. *Considering a neuron i, if its activation magnitude h_i^c over the data of class c is larger than the activation $h_i^{c'}$ over data of other classes c', i.e., $\min_{i \in \mathbb{D}^c} h_i^c > \max_{i \in \mathbb{D}^{c'}} h_i^{c'}, \forall c' \in [C], c' \neq c$, then the neuron i is defined as the bound neuron of the class c.*

Now, based on the theories of neural collapse [6, 10, 28], we show that each data class corresponds to a subset of bounded neurons in converged neural networks by the following theorem.

Theorem 1. *Considering the neural network has reached the optimal solution of (1) over the balanced-class dataset, if the loss function $f(\cdot)$ is cross-entropy (CE) or mean-square-error (MSE) and the activation function is ReLU, then there exists a non-empty subset $\mathcal{H}_c$ consisting of bound neurons for each class c.*

Proof. We denote the parameters of the last classifier layer corresponding to the class c as $\mathbf{w}_{L,c}$, and denote the activation of the last-second layer over the data of class c as $\mathbf{h}_{L-1}^c$. Based on the theories of neural collapse, it can be concluded that

$$\mathbf{w}_{L,c} = A\mathbf{h}_{L-1}^c, \tag{2}$$

when the model achieves the optimal solution for CE loss (Theorem 1 of [6]) and for MSE loss (Theorem 3 of [10]), where $A > 0$ is a constant. Since the optimal model minimizes the objective function (1), it makes correct predictions over the training dataset. Hence, for any data sample of the class c, its prediction score p_c corresponding to the class c is larger than other classes c', i.e., $p_c > p_{c'}$. The prediction score p_c for each class c is calculated based on the softmax $p_c = \frac{\exp(\mathbf{w}_{L,c}^T \mathbf{h}_{L-1}^c / T)}{\sum_{i=1}^{C} \exp(\mathbf{w}_{L,i}^T \mathbf{h}_{L-1}^c / T)}$. Based on this definition, the prediction scores of different classes for the sample x of class c satisfy

$$p_c = \frac{\exp(\mathbf{w}_{L,c}^T \mathbf{h}_{L-1}^c / T)}{\sum_{i=1}^{C} \exp(\mathbf{w}_{L,i}^T \mathbf{h}_{L-1}^c / T)} > p_{c'} = \frac{\exp(\mathbf{w}_{L,c'}^T \mathbf{h}_{L-1}^c / T)}{\sum_{i=1}^{C} \exp(\mathbf{w}_{L,i}^T \mathbf{h}_{L-1}^c / T)} \tag{3}$$

for any class $c' \in [C], c' \neq c$. By eliminating the denominators of both sides of (3), we get that

$$\mathbf{w}_{L,c}^T \mathbf{h}_{L-1}^c > \mathbf{w}_{L,c'}^T \mathbf{h}_{L-1}^c. \tag{4}$$

Bringing (2) to (4), we have

$$A(\mathbf{h}_{L-1}^c)^2 > (\mathbf{h}_{L-1}^{c'})^T \mathbf{h}_{L-1}^c, \tag{5}$$

which directly derives the following inequality

$$(\mathbf{h}_{L-1}^c - \mathbf{h}_{L-1}^{c'})^T \mathbf{h}_{L-1}^c > 0 \Leftrightarrow \sum_{i=1}^{M_{L-1}} (h_{L-1,i}^c - h_{L-1,i}^{c'}) h_{L-1,i}^c > 0, \tag{6}$$

which indicates that there exists a subset of neurons $\mathcal{H}^c \subset [M_{L-1}]$, the following inequality holds

$$(h_{L-1,i}^c - h_{L-1,i}^{c'}) h_{L-1,i}^c > 0. \tag{7}$$

for each $i \in \mathcal{H}^c$. Since the activation output by the ReLU function is always non-negative, i.e., $h^c_{L-1,i} \geq 0, \forall i \in [M_{L-1}]$, we can immediately obtain that

$$h^c_{L-1,i} - h^{c'}_{L-1,i} > 0. \tag{8}$$

for each $i \in \mathcal{H}^c$. The proof is done. □

Noting that the activation function is $\mathrm{ReLU}(\mathbf{w}^T_{l,i}\mathbf{h}_{l-1})$ and the prediction probability p_c for each class c is $p_c = \frac{\exp(\mathbf{w}^T_{L,c}\mathbf{h}^c_{L-1}/T)}{\sum_{i=1}^C \exp(\mathbf{w}^T_{L,i}\mathbf{h}^c_{L-1}/T)}$, indicating that high activation magnitude strongly contributes to the classification result. In the following text, for ease of expression, we also call the neurons subset $\mathcal{H}^c$ as the bound neurons of the class c. Obviously, reducing the activation magnitude of bound neurons corresponding to some given class c will inevitably decrease its prediction probability and thus decrease the prediction accuracy. Unfortunately, in FL scenario, we show that the activation magnitude of the bound neurons of some data classes may be severely reduced during the training process due to the data heterogeneity across clients. We show this by specifying the following example.

Example 1. We consider training a two-layer neural network over two clients, as shown in Figure 1. The task is the binary classification with the label $y = 1$ or $y = 2$ corresponding to samples distributed around $(1, 1/3)$ and around $(1/3, 1)$ respectively. By simulating the data heterogeneity, we consider each client has only access to the samples of one single class ($y = 1$ for client 1 and $y = 2$ for client 2). The initialized parameters of the neural network are presented in the server of Figure 1a. The bound neurons for the class $y = 2$ are denoted by gold color, i.e., the second hidden neuron $h_{1,2}$ which has a high activation over the data of class 2. Now, we show that the expected activation of all neurons over the class 2 is reduced after local training in client 1:

$$\begin{aligned}
\Delta h_{1,1} &= h^2_{1,1} - \tilde{h}^2_{1,1} = 0.67 - 0.73 = -0.06,\\
\Delta h_{1,2} &= h^2_{1,2} - \tilde{h}^2_{1,2} = 1.11 - 1.04 = 0.07,\\
\Delta p_1 &= p_1 - \tilde{p}_1 = 0.45 - 0.63 = -0.18,\\
\Delta p_2 &= p_2 - \tilde{p}_2 = 0.55 - 0.37 = 0.18,
\end{aligned} \tag{9}$$

where $h^2_{1,2}$ and $\tilde{h}^2_{1,2}$ represent the activation of class 2 before and after training an iteration in client 1 with the learning rate 0.5. p_2 and p'_2 represent the probability score over the data of the class 2 before and after local training. By observing the value $\Delta h_{1,2}$, we can conclude that the activation of bound neurons over class 2 is greatly reduced after local training in client 1. This also leads to a probability reduction as presented by Δp_2, decreasing the accuracy of the class 2. As a comparison, the values of $\Delta h_{1,1}$ and Δp_1 indicate that the risk of class 2 data being misclassified into class 1 increases. We call this phenomenon *neuron drift*, which is the essence of neural networks occurring *client drift* [1, 17].

The example shows that each client tends to train the neural network to maximize the probability score over its local data. The client realizes its objective by increasing the activation magnitude of neurons over its local data classes. When the bound neurons of a class missed by the client are included in these updated neurons, the phenomenon of neuron drift occurs. In fact, the phenomenon of neuron drift presented in Example 1 exists in general two-layer neural networks.

Theorem 2. *Considering a two-layer neural network has reached the optimal solution of (1) over the balanced-class dataset of the binary classification task, and its activation function is ReLU, if the neural network is trained a local step in a client k without the data of class c, then the activation magnitude of the bonded neurons $\mathcal{H}_c$ over the class c data $x^c_i \in \mathbb{D}^c$ are reduced as $x^T_k x^c_i > 0$. More specifically, the activation reduction $\Delta h_{1,m}$ of the m-th neuron of the hidden layer is $\mathbf{w}^T_{1,m}x^c_i$ as $\mathbf{w}^T_{1,m}x^c_i - \eta \sum_{x_k \in \mathbb{D}_k} x^T_k x^c_i \big((p_{c'}(x_k) - 1)w_{2,c',m} + p_c(x_k)w_{2,c,m}\big) \leq 0$, and otherwise, the activation reduction is $\eta \sum_{x_k \in \mathbb{D}_k} x^T_k x^c_i \big((p_{c'}(x_k) - 1)w_{2,c',m} + p_c(x_k)w_{2,c,m}\big)$.*

Proof. Consider the class c and one of its bonded neurons $h_{1,m} \in \mathcal{H}^c$ in the hidden layer. The updated formula of the parameter connected to this neuron is:

$$\tilde{w}_{1,m,j} = w_{1,m,j} - \eta \sum_{x_k \in \mathbb{D}_k} x^j_k \big((p_{c'}(x_k) - 1)w_{2,c',m} + p_c(x_k)w_{2,c,m}\big),$$

where $\tilde{w}_{1,m,j}$ denotes the parameter after local training in the client k, $p_c(x_k)$ denotes the probability score of x_k over the class c, and x^j_k denotes the j-th dimension of the sample x_k. Considering the model has achieved convergence of neural collapse, based on equation (2), i.e., $\mathbf{w}_{2,c} = A\mathbf{h}^c_1, \forall c \in [C]$, we have

$$\begin{aligned}
&(p_{c'}(x_k) - 1)w_{2,c',m} + p_c(x_k)w_{2,c,m} \\
&= -p_c(x_k)w_{2,c',m} + p_c(x_k)w_{2,c,m} = p_c(x_k)(Ah^c_{1,m} - Ah^{c'}_{1,m}) > 0,
\end{aligned} \tag{10}$$

where the last inequality is based on the Definition 1 for the bonded neurons. Now, for each data sample $x^c_i \in \mathbb{D}^c$ of the class c, when $x^T_k x^c_i > 0$ for all sample $x_k \in \mathbb{D}_k$ of client k, we can compute the change of the activation $h_{1,m}$ of the bonded neuron before and after local updating in the client k:

$$\begin{aligned}
&\Delta h_{1,m} = \mathrm{ReLU}(\mathbf{w}^T_{1,m}x^c_i) - \mathrm{ReLU}(\tilde{\mathbf{w}}^T_{1,m}x^c_i) = \mathrm{ReLU}(\mathbf{w}^T_{1,m}x^c_i)\\
&-\mathrm{ReLU}\Big(\mathbf{w}^T_{1,m}x^c_i - \eta \sum_{x_k \in \mathbb{D}_k} x^T_k x^c_i \big((p_{c'}(x_k) - 1)w_{2,c',m} + p_c(x_k)w_{2,c,m}\big)\Big)\\
&\geq 0,
\end{aligned} \tag{11}$$

where d is the dimension of the data space. The last inequality is derived from (10). More specifically, as $\mathbf{w}^T_{1,m}x^c_i - \eta \sum_{x_k \in \mathbb{D}_k} x^T_k x^c_i \big((p_{c'}(x_k) - 1)w_{2,c',m} + p_c(x_k)w_{2,c,m}\big) \leq 0$, the activation reduction is

$$\Delta h_{1,m} = \mathrm{ReLU}(\mathbf{w}^T_{1,m}x^c_i) - 0 = \mathbf{w}^T_{1,m}x^c_i, \tag{12}$$

and otherwise,

$$\begin{aligned}
&\Delta h_{1,m}\\
&= \mathbf{w}^T_{1,m}x^c_i - \mathbf{w}^T_{1,m}x^c_i + \eta \sum_{x_k \in \mathbb{D}_k} x^T_k x^c_i \big((p_{c'}(x_k) - 1)w_{2,c',m} + p_c(x_k)w_{2,c,m}\big)\\
&= \eta \sum_{x_k \in \mathbb{D}_k} x^T_k x^c_i \big((p_{c'}(x_k) - 1)w_{2,c',m} + p_c(x_k)w_{2,c,m}\big).
\end{aligned} \tag{13}$$

The proof is done. □

Although our formal analysis is only based on the two-layer neural network, it also provides some insights into the deeper neural networks. Intuitively, we can view the deep neural network as a series of two-layer neural networks. Except for the last two-layer neural network, which uses real labels as supervision, all others use the activation of the intermediate layer as supervision. In fact, existing studies have shown that the separability of activation between classes also exists and progressively increases from shallow to deep layers at a linear rate [11, 15, 34], which indicates that these intermediate activation can be viewed as variants of real label. As a consequence, the neuron drift may also exist in deep neural networks, but in different degrees in different layers. The deeper the layer, the more serious the neuron drift. This intuition is also consistent with the experimental observations in [30], where activation similarity among clients linearly reduces with the growth of layers. Experimental verfications can be found in Appendix A.

5 Methodology

In this section, we propose a simple yet effective method named FedNLR to mitigate the neuron drift. Specifically, the core idea is to adopt neuron-level learning rates during the local training process. The algorithm workflow is presented in Algorithm 1.

As analyzed above, the activation of bound neurons over some classes may be reduced after training the neural network in the client without the data of these classes. The principle behind this is that the client tends to increase the activation of these bound neurons over its local data. To solve this problem, a direct way is to identify the bound neurons using these missed classes and then, set small learning rates for the parameters of these neurons to restrict their activation reduction.

However, such a way is impractical because the client has no access to the data of these missed classes. Considering this, we propose an approximation algorithm, which sets learning rates according to the activation magnitude of the global model over the local data. The learning rates of parameters of low-activation neurons are set to be small. The principle is that the bound neurons are included in the set of neurons with low activation magnitude over local data. Therefore, restricting them can also mitigate the activation reduction of bound neurons over missed data classes.

Consider the local training process of client k in each round t. To adapt the neuron-level learning rates, before performing local training steps, each client first computes the activation $h_{l,m}$ for each neuron of the received global model $\mathbf{w}_t$ over its local dataset $\mathbb{D}_k$ and then computes the activation average $\bar{h}_{l,m}$ over all samples:

$$\bar{h}_{l,m} = \frac{1}{D_k} \sum_{x_i \in \mathbb{D}_k} h_{l,m}(x_i). \tag{14}$$

Since the activation values of neurons in different layers are different, the client sets the learning rates for neurons layer by layer. Specifically, for each layer l, the learning rate $\hat{\eta}_{l,m}$ of the m-th neuron parameters is set to be:

$$q_{l,m} = \frac{e^{\bar{h}_{l,m}/T_l}}{\sum_{i=1}^{M_l} e^{\bar{h}_{l,i}/T_l}}, \quad \hat{\eta}_{l,m} = \eta \cdot \frac{M_l q_{l,m}}{\sum_{i=1}^{M_l} q_{l,m}}, \tag{15}$$

where η is the basic learning rate used in existing methods and $q_{l,m}$ denotes the scale of the learning rate. The temperature parameter T_l controls the discrepancy between the maximized and the minimized learning rate for the neurons of the l-th layer. A larger T_l indicates a larger discrepancy. Specifically, to avoid excessively high or low learning rates, we use a hyper-parameter $\mu_l = \frac{\hat{\eta}_{l,\max}}{\hat{\eta}_{l,\min}}$ to be the ratio between the maximized and the minimized learning rate for the neurons of the l-th layer, where $\hat{\eta}_{l,\max} = \max(\hat{\eta}_{l,1}, \ldots, \hat{\eta}_{l,M_l})$ and $\hat{\eta}_{l,\min} = \min(\hat{\eta}_{l,1}, \ldots, \hat{\eta}_{l,M_l})$. Then, we can obtain

Algorithm 1: Algorithm workflow of FedNLR

```
Input :$T$: communication round; $K$: client number; $\eta$:
       basic learning rate; $\mu_0, a_1, a_2$: discrepancy rate
       between maximum and minimum learning rates;
1  Initialize the parameter $\mathbf{w}^1$;
2  for $t = 1$ to $T$ do
3      Randomly select $K_t$ clients and send the global model
       $w^t$ to them;
4      for each selected client $k$ in parallel do
5          Compute activation average $\bar{h}_{l,m}$ with (14);
6          Compute the discrepancy ratio $\mu_l$ with (17);
7          Compute the temperature $T_l$ with (16);
8          Compute the learning rate $\eta_{l,m}$ via (15);
9          Update local model $\mathbf{w}_k$ for $E$ iterations with (18);
10         Send the model $\mathbf{w}_k^{t,E}$ to the server;
11     end
12     Aggregate local models with (19);
13 end
```

$$\mu_l = \frac{\hat{\eta}_{l,\max}}{\hat{\eta}_{l,\min}} = e^{(\bar{h}_{l,\max} - \bar{h}_{l,\min})/T_l} \Rightarrow T_l = \frac{\bar{h}_{l,\max} - \bar{h}_{l,\min}}{\ln(\mu_l)}. \tag{16}$$

Setting μ_l for each layer of the neural network requires significant human costs, we propose a heuristic formula to make configurations for all layers as:

$$\mu_l = \mu_0 + a_1 \cdot l/L + a_2 \cdot \log M_l, \tag{17}$$

where μ_0, a_1, and a_2 are positive constants. The configuration of μ_l is mainly based on the *depth* and the *width* of the neural network layer. The intuition behind this design is based on the sharing degree of high-activation neurons between different data classes. Even though the bound neurons of some missed classes are reduced by the client, these classes can still achieve high probability scores with the shared neurons. Hence, the ratio is lower to relax the restriction when different classes share more high-activation neurons. Generally, shallow layers reveal the common knowledge shared among classes and they may activate the similar neurons. Conversely, deep layers extract different knowledge from different classes, and each class tends to activate distinct neurons. Therefore, we set lower ratios for shallow layers. A similar case also exists in the layer width. When the number of neurons in a layer is small, these classes have to share them to improve their probability scores. Hence, we set smaller ratios for narrow layers.

Finally, with the learning rate $\eta_{l,m}$ for the parameters $\mathbf{w}_{l,m}$ of each m-th neuron of the l-th layer, the client k runs update:

$$\mathbf{w}_{l,m}^k = \mathbf{w}_{l,m}^k - \hat{\eta}_{l,m} \nabla_{\mathbf{w}_{l,m}^k} f_k(\mathbf{w}^k), \tag{18}$$

where $\nabla_{\mathbf{w}_{l,m}^k} f_k(\mathbf{w}^k)$ denotes the gradient over a mini-batch of samples randomly sampled from the local dataset $\mathbb{D}_k$. After performing E local iterations, the client k uploads the locally updated model $\mathbf{w}^k$ to the server, and the server aggregates received local models into the global model $\mathbf{w}$:

$$\mathbf{w} = \sum_{k=1}^{K_t} \frac{D_k}{\sum_{k=1}^{K_t} D_k} \mathbf{w}^k, \tag{19}$$

where K_t is the number of the t-th round selected clients.

6 Theoretical Analysis

Besides, we make the following assumptions for these objectives which are widely adopted in FL [5, 40].

Assumption 1. ***(L-smoothness)****. The objective function F_n is L-smooth with Lipschitz constant $L > 0$, i.e., $\|\nabla F_n(\mathbf{w}) - \nabla F_n(\mathbf{w}')\|_2 \le L\|\mathbf{w} - \mathbf{w}'\|_2$ for all $\mathbf{w}, \mathbf{w}'$.*

Assumption 2. ***(Bounded Variance)****. For all parameters $\mathbf{w}$, the variance of the local stochastic gradient in each client is bounded by σ_l^2: $\mathbb{E}(\|\nabla f_n(\mathbf{w}) - \nabla F_n(\mathbf{w})\|^2) \le \sigma_l^2$. Besides, the norm of gradient is bounded by M: $\|\nabla F_n(\mathbf{w})\|^2 \le M^2$.*

Based on the above assumptions, we have the following theory for the convergence of the proposed algorithm.

Theorem 3. *Consider problem (1) under Assumption 1 and 2. If the learning rate η diminishes with $O(\sqrt{\frac{K}{T}})$, then the global model $\mathbf{w}_t$ obtained by Algorithm 1 achieves asymptotic convergence, i.e.,*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla F(\mathbf{w}_t)\|_2^2 \le \frac{2(F(\mathbf{w}_1) - F_*))\sqrt{\mu_{max}(M_{max} - 1) + 1}}{\sqrt{KT}}$$
$$+ \frac{\mu_{max}^2(\mu_{max}(M_{max} - 1) + 1)\sigma_l^2 L}{(\mu_{max} + M_{min} - 1)^2 r\sqrt{KT}} + \frac{4\mu_{max}^3(\mu_{max}(M_{max} - 1) + 1)^2 E^2 M^2}{(\mu_{max} + M_{min} - 1)^3 \sqrt{T}}$$
$$+ \frac{(1 - r)\mu_{max}^2(\mu_{max}(M_{max} - 1) + 1)M^2 L}{(\mu_{max} + M_{min} - 1)^2 r\sqrt{T}}, \tag{20}$$

where r is the ratio of participated clients in each round.

With Theorem 3, we ensure the convergence of the global model. Although the convergence rate does not receive improvement over baselines, empirical results demonstrate the effectiveness of our method. The details of the proof can be found in Appendix B.

7 Experiment

In this section, we evaluate the accuracy of FedNLR and compare it with several advanced methods in various datasets and NonIID settings. Due to the page limitation, more details are available in the supplementary materials.

7.1 Setup

Dataset. We explore 3 benchmark datasets: CIFAR-10[20], CIFAR-100[20], CINIC[4]. For both of them, we use the two NonIID data settings: the Dirichlet distribution[46] and the shards-based segmentation [33]. In the *Dirichlet* (n, α) function, smaller α leads to more NonIID level. In the *Shards* (n, S) function, S represents the number of classes in one client. For example, *Shards* (20, 2) with CIFAR-10 dataset will allocate each client with 2 classes of images. Notice that both functions are unbalanced, allocating un-uniform size of data across clients. Furthermore, we choose different scenarios of data distribution by setting α and S in [2, 4, 6, 8] to simulate different levels of real-world NonIID scenarios.

Baseline. Baselines include FedAvg[32], FedProx[24], Scaffold[17], FedNova[40], FedRS[26], FedLC[47], FedDyn[1], CCVR[30].

Hyper-parameter Settings. The hyper-parameters for each method were set according to their respective original papers. For instance, the hyper-parameter for FedProx is set at 0.01, for FedDyn is 0.1, for FedLC is 1, for FedRS is 0.5, and for FedNLR, it is self-adjusted using an empirical formula as outlined in Eq. (17). The fine-tuned empirical formula dynamically adjusts μ_l based on layer depth and neuron count. The process of fine-tuning the specific factors is further explained in Sec. 7.2.

Configurations. Unless otherwise mentioned, we set the number of local training epoch $E = 2$, communication round $T = 500$, number of clients $n = 20$, participation ration 0.4 in each round; batch size 64, learning rate $\eta = 0.01$. We develop a customized Stochastic Gradient Descent (SGD) optimizer, capable of handling vector-form learning rates. All methods run without momentum or weight decay, except FedDyn requiring a weight decay of 0.1 and running 200 rounds. These experiments are run on the hardware of 4 NVIDIA GeForce RTX 3090 GPUs and the software framework of PyTorch 2.0. Each experimental setting is run twice, with the final 5 rounds' ACC(accuracy) averaged and standard variance calculated. We employ VGG-9 [21] as the basic model architecture, initialized using the Kaiming uniform function for convolutional layers and the Xavier normal function for linear layers.

7.2 Result and Analysis

Better Performance. As shown in Tab. 1 and Tab. 2, under all NonIID conditions, FedNLR or "FedNLR + FedDyn" demonstrates the best performance on both datasets. Although in a few cases, FedNLR's performance is not as good as FedDyn's, the combination with FedDyn leads to significant improvement, achieving the optimal state. In the CIFAR-100-*Dirichlet(*20, 0.05*)*, FedNLR achieves a 3.9% promotion compared with FedAvg, while other methods only raise around 1%. After enough rounds of training, other methods don't show an improvement because they are not designed to focus on convergence accuracy. Even FedDyn has made great improvements in the CIFAR-10-*Dirichlet(*20, 0.5*)*, the addition of FedNLR still shows gains of 1.1%. Moreover, the improvements of FedNLR grow with the degree of NonIID, as proved by the CIFAR-100-*Dirichlet(*20, 0.05*)* and the less NonIID CIFAR-100-*Dirichlet(*20, 0.5*)*.

Robustness on heterogeneous data. We find that FedNLR exhibited greater performance improvements not only in situations with higher NonIID degrees but also in a larger number of categories, inferring from the comparison of the CIFAR-100-*Dirichlet(*20, 0.05*)* and the fewer categories of CIFAR-10-*Dirichlet(*20, 0.05*)*. This indicates that FedNLR has significant advantages in handling different types and complexities of data distributions. On the other hand, we have run the experiment settings in different learning rates as shown in Fig. 2, in which the resulting improvements are not affected by the learning rate. Furthermore, when the client participation ratio decreases, the NonIID influences more strongly because of the unbalanced local datasets. We use 100 clients and choose 10% participation in each round and calculate the average and variance of the last 30 rounds. As shown in Tab. 3, FedNLR presents a consistent improvement across different NonIID levels. In the CIFAR-10-*Shards(*100, 4*)*, FedDyn improves 4.57% and can

Table 1: The comparison of final test accuracy on the two datasets. The best result in each setting is bolded.

Method	CIFAR-10(%)				CIFAR-100(%)			
Shards (n, S)	(20, 2)	(20, 4)	(20, 6)	(20, 8)	(20, 20)	(20, 40)	(20, 60)	(20, 80)
FedAvg	72.11±0.61	78.10±0.52	81.26±0.29	81.38±0.19	45.51±0.59	49.40±0.13	49.64±0.45	50.44±0.09
FedProx	71.78±0.32	78.90±0.28	80.99±0.17	81.42±0.14	45.17±0.37	48.43±0.34	49.53±0.36	50.71±0.06
Scaffold	71.76±0.20	78.11±0.46	81.19±0.15	81.13±0.34	45.64±0.32	48.92±0.08	49.78±0.20	51.81±0.32
FedNova	71.64±0.28	77.82±0.33	81.94±0.12	81.61±0.16	46.02±0.44	48.80±0.20	49.88±0.25	50.48±0.39
FedRS	72.13±0.82	78.53±0.28	81.99±0.15	80.95±0.17	46.21±0.59	49.41±0.17	50.75±0.37	51.34±0.19
FedLC	71.24±0.18	78.30±0.33	81.53±0.24	81.74±0.27	46.04±0.53	49.35±0.20	50.24±0.16	51.60±0.24
FedDyn	68.33±1.52	80.65±0.59	84.71±0.40	85.35±0.21	28.88±0.26	31.18±0.30	31.32±0.23	34.09±0.20
FedNLR	**74.11**±0.48	80.10±0.37	82.77±0.26	82.24±0.12	**47.93**±0.49	**50.20**±0.28	**51.79**±0.62	**52.46**±0.17
FedDyn + FedNLR	73.27±0.33	**81.93**±0.30	**85.67**±0.26	**85.89**±0.08	35.03±0.15	33.86±0.28	33.44±0.42	35.99±0.30
Dirichlet (n, α)	(20, 0.05)	(20, 0.1)	(20, 0.3)	(20, 0.5)	(20, 0.05)	(20, 0.1)	(20, 0.3)	(20, 0.5)
FedAvg	65.42±1.14	72.66±0.20	79.05±0.20	80.13±0.24	40.71±0.55	44.12±0.15	48.89±0.44	50.62±0.55
FedProx	64.02±2.04	72.51±0.76	78.49±0.20	80.20±1.27	41.10±0.62	43.82±0.31	48.49±0.18	50.14±0.77
Scaffold	63.22±3.49	71.84±0.48	78.94±0.17	79.86±0.66	41.86±0.19	44.76±0.63	49.20±0.61	50.26±0.36
FedNova	63.14±1.08	72.58±0.44	79.13±0.42	79.30±2.67	41.49±0.51	44.00±0.48	49.40±0.18	50.72±0.21
FedRS	64.98±0.81	71.54±0.26	78.70±0.21	80.07±0.27	41.25±0.55	44.91±0.36	49.53±0.23	51.26±0.17
FedLC	65.06±1.07	72.31±0.38	78.92±0.23	80.95±0.28	41.74±0.30	44.83±0.59	49.10±0.41	50.20±0.62
FedDyn	51.06±1.22	72.15±0.64	83.26±0.13	84.70±0.30	29.09±0.31	30.64±0.18	33.10±0.19	35.57±0.27
FedNLR	**66.11**±0.89	73.92±0.60	80.61±0.11	81.65±0.22	**44.67**±0.48	**47.14**±0.48	**51.29**±0.28	**51.77**±0.15
FedDyn + FedNLR	48.29±2.14	**75.02**±0.48	**84.17**±0.36	**85.82**±0.31	34.37±0.73	34.78±0.48	35.59±0.33	36.52±0.25

Table 2: The accuracy on the CINIC dataset.

Method	Shards(S)		Dir(α)	
	2	4	0.05	0.1
FedAvg	50.88	53.03	43.37	47.51
FedProx	52.12	52.75	43.17	46.74
Scaffold	52.07	52.42	43.33	46.91
FedNova	51.07	52.54	41.55	46.63
FedRS	51.19	52.13	44.47	46.72
FedLC	51.49	52.81	43.64	47.67
CCVR	50.59	51.31	42.66	46.93
FedDyn	47.56	55.99	42.83	50.22
FedNLR	**52.17**	52.30	43.37	47.81
FedDyn+FedNLR	49.49	**56.24**	**44.69**	**51.93**

gain an additional 2.91% when integrated with FedNLR. Combining FedDyn with FedNLR performs the best in these NonIID situations.

Orthogonal Combination with other algorithms. FedNLR can be divided into two methods: the aggregation method, which can be discarded, and the learning rate scheduler, which can be integrated with many other methods, such as FedRS, FedLC, FedNova, and FedDyn. With only a few lines of code, the customized SGD is capable of handling a vector-form learning rate for one layer. Using the customized SGD, other methods are compatible with the learning rate scheduler. The integration of FedDyn with FedNLR demonstrates the capability to achieve state-of-the-art accuracy, particularly under conditions of low Non-IID levels. This suggests a synergistic effect between the two methods, enhancing model performance in scenarios characterized by more homogeneous data distributions. Because the accuracy of a model is directly correlated with the utility of the generated activation map: a more accurate model yields a more informative activation map. Consequently, a

Table 3: 10% clients participate in each round. The comparison of final test accuracy on the two datasets.

Method	CIFAR-10(%)	
Shards (n, S)	(100, 4)	(100, 8)
FedAvg	72.94±1.33	75.03±1.27
FedProx	72.33±1.28	75.27±1.19
Scaffold	72.60±1.35	75.63±0.97
FedNova	72.61±1.44	74.65±1.13
FedRS	72.83±1.16	74.88±1.00
FedLC	72.72±1.25	74.32±1.07
FedDyn	77.51±1.84	83.32±0.51
FedNLR	73.38±1.48	76.14±1.31
FedDyn + FedNLR	**80.42**±1.59	**85.26**±0.57
Dirichlet (n, α)	(100, 0.1)	(100, 1)
FedAvg	63.30±2.31	77.83±0.95
FedProx	62.42±2.37	77.03±1.71
Scaffold	61.79±2.34	77.27±0.71
FedNova	55.49±5.80	78.21±0.97
FedRS	63.58±2.42	77.98±0.92
FedLC	63.17±2.59	77.97±0.66
FedDyn	63.98±2.39	85.22±0.34
FedNLR	63.40±2.51	78.73±1.47
FedDyn + FedNLR	**64.16**±2.37	**86.10**±0.39

learning rate scheduler derived from this enhanced activation map can provide more insightful guidance for model training.

7.3 Hyper-Parameter Sensitivity Analysis and Ablation Study

Scale Factor Normalization with Range Control. The purpose

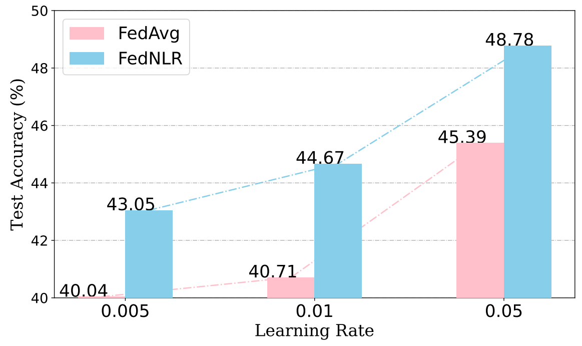


Figure 2: Results of FedNLR using different learning rates with CIFAR-100-*Dirichlet*(20, 0.05).

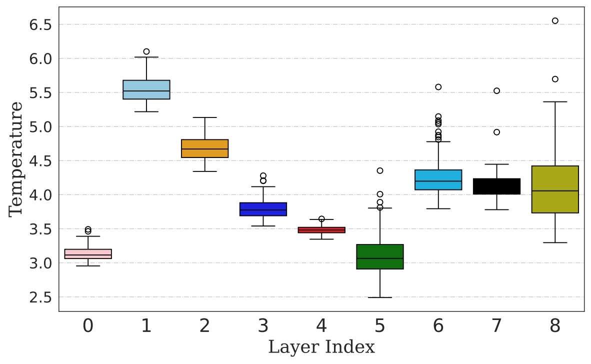


Figure 3: The temperature T_l for each layer l.

of the μ is to limit the scale factor for learning rates of a weighted layer in an interval where the largest scale factor equals the multiplication of μ and the smallest scale factor. Also, the average value of the scale factor in one layer is 1. As depicted in Fig. 3, the temperature of the SoftMax function is used for the scale factor normalization. The variance of temperature shows the crucial role of μ in reducing manual setting efforts. Empirically, the second layer typically requires the highest temperature, because it consists of the most neurons across all layers.

Ablation Experiment of Scaling Each Layer. Fig. 4 shows the ablation experiment results, proving each layer's performance improvement. Most points are above the red line of FedAvg, proving the hyperparameter's insensitivity. There is not a constant number capable of continuously improving performance, but in most situations, 1.75 and 2 show improvements. And the hidden layers' variance is bigger than the shallow layers and the classifier. This is evidence that hidden layers' neuron values vary most concerning the change of classes. Intuitively, if we simply sum up all the improvements of all layers with manually chosen μ, then the final accuracy should be best. However, Fig. 5 proves that the setting of the math formula Eq. (17) calculated μ is better, while manually chosen μ doesn't perform well due to the interaction attribute among all layers inside neural networks. Also, if we only consider the width or depth, then the performance is not comparable with the best result. Owing to constraints in computational power, we have identified an optimal combination of hyper-parameters within a limited range. As the two parameters increase from 1 to

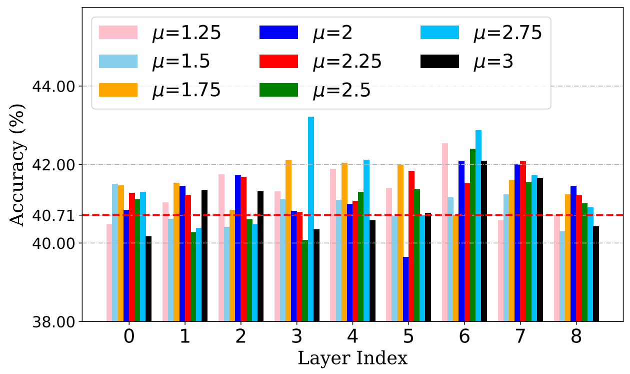


Figure 4: Experimental result of 9 layers, with individual hyper-parameter μ, under the setting of CIFAR-100-*Dirichlet*(20, 0.05).

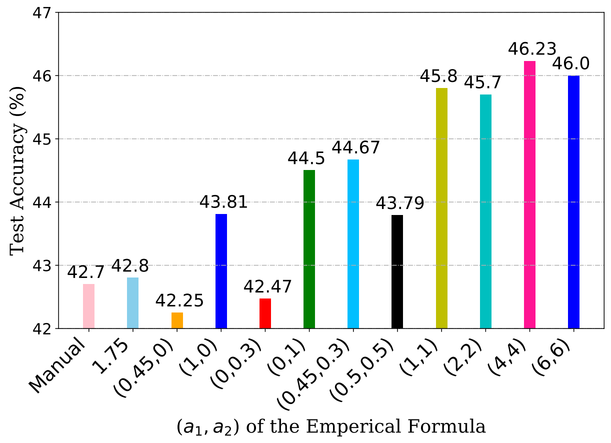


Figure 5: Performance improvements of FedNLR with different μ under the setting of CIFAR-100-*Dirichlet*(20, 0.05).

4 and 6, the average accuracy increases to 46.23, and then slightly decreases, but the overall performance is still optimal. This may indicate that the moderate increase of depth and width parameters can improve the model performance, but the effect will decrease after exceeding a certain range. The main experiment was initially conducted using (0.45, 0.3). However, potential improvements are expected if this is changed to (1, 1). Consequently, we configure the equation Eq. (17) as the following empirical mathematical formula: $\mu_l = 1 + \frac{l}{L} + \log_{10}(M_l)$, which also applies to other models simply.

8 Conclusion

This paper conducts research on the problem of model drift in federated learning *for training deep neural networks.* We provide a deep understanding of the formation process of model drift. Specifically, instead of viewing the neural network as a whole, we decompose it into a set of neurons and track the changes of each neuron during local training. We identify that each class is bound to a specific set of neurons and their activation can be reduced during the local training process when this class is not included by the client. Based on this motivation, we propose a simple yet effective method named FedNLR which adopts neuron-wise learning rates to mitigate the drift of neurons. Our theoretical results guarantee the convergence of FedNLR and empirical results also demonstrate its effectiveness.

Despite the analysis for understanding the evolution of specific neurons, this paper is limited to the two-layer neural network which has great potential to be unleashed. Recently, many explanation methods for deep neural networks have been proposed, which may shed light on expand the analysis. For example, we may utilize the Shapley value to identify accurate bound neurons to specific samples. We leave them as the future work.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China under grants 62302184, 62376103, 62206102, Science and Technology Support Program of Hubei Province under grant 2022BAA046, Ant Group through CCF-Ant Research Fund,and two grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU15222621, PolyU15225023). Besides, the research is supported under the National Key R&D Program of China (2022ZD0160201) and the RIE2020 Industry Alignment Fund - Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contributions from the industry partner(s).

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. 2021. Federated Learning Based on Dynamic Regularization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[2] Muhammad Ammad-ud-din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. 2019. Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System. *CoRR* abs/1901.09888 (2019).

[3] Zachary Charles and Jakub Konečný. 2021. Convergence and Accuracy Trade-Offs in Federated Learning and Meta-Learning. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*. 2575–2583.

[4] Luke Nicholas Darlow, Elliot J. Crowley, Antreas Antoniou, and Amos J. Storkey. 2018. CINIC-10 is not ImageNet or CIFAR-10. *CoRR* abs/1810.03505 (2018).

[5] Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. 2020. Personalized Federated Learning with Moreau Envelopes. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*.

[6] Cong Fang, Hangfeng He, Qi Long, and Weijie J Su. 2021. Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. *Proceedings of the National Academy of Sciences* 118, 43 (2021), e2103091118.

[7] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. 2022. FedDC: Federated Learning with Non-IID Data via Local Drift Decoupling and Correction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24*. 10102–10111.

[8] Yonghai Gong, Yichuan Li, and Nikolaos M. Freris. 2022. FedADMM: A Robust Federated Deep Learning Framework with Adaptivity to System Heterogeneity. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. 2575–2587.

[9] Pengfei Guo, Puyang Wang, Jinyuan Zhou, Shanshan Jiang, and Vishal M. Patel. [n. d.]. Multi-Institutional Collaborations for Improving Deep Learning-Based Magnetic Resonance Image Reconstruction Using Federated Learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. 2423–2432.

[10] X. Y. Han, Vardan Papyan, and David L. Donoho. 2022. Neural Collapse Under MSE Loss: Proximity to and Dynamics on the Central Path. In *The Tenth International Conference on Learning Representations, ICLR, Virtual, April 25-29, 2022*.

[11] Hangfeng He and Weijie J. Su. 2022. A Law of Data Separation in Deep Learning. *CoRR* abs/2210.17020 (2022).

[12] Ming Hu, Yue Cao, Anran Li, Zhiming Li, Chengwei Liu, Tianlin Li, Mingsong Chen, and Yang Liu. 2024. FedMut: Generalized Federated Learning via Stochastic Mutation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 12528–12537.

[13] Ming Hu, Zeke Xia, Dengke Yan, Zhihao Yue, Jun Xia, Yihao Huang, Yang Liu, and Mingsong Chen. 2023. GitFL: Uncertainty-Aware Real-Time Asynchronous Federated Learning Using Version Control. In *In Proceedings of IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 145–157.

[14] Ming Hu, Peiheng Zhou, Zhihao Yue, Zhiwei Ling, Yihao Huang, Anran Li, Yang Liu, Xiang Lian, and Mingsong Chen. 2024. FedCross: Towards Accurate Federated Learning via Multi-Model Cross-Aggregation. In *IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2137–2150.

[15] Like Hui, Mikhail Belkin, and Preetum Nakkiran. 2022. Limitations of Neural Collapse for Understanding Generalization in Deep Learning. *CoRR* abs/2202.08384 (2022).

[16] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. 2023. FedExP: Speeding Up Federated Averaging via Extrapolation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5*.

[17] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. [n. d.]. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. 5132–5143.

[18] Geeho Kim, Jinkyu Kim, and Bohyung Han. 2022. Communication-Efficient Federated Learning with Acceleration of Global Momentum. *CoRR* abs/2201.03172 (2022).

[19] Jinkyu Kim, Geeho Kim, and Bohyung Han. [n. d.]. Multi-Level Branched Regularization for Federated Learning. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. 11058–11073.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[21] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated Learning on Non-IID Data Silos: An Experimental Study. In *ICDE*. IEEE, 965–978.

[22] Qinbin Li, Bingsheng He, and Dawn Song. [n. d.]. Model-Contrastive Federated Learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. 10713–10722.

[23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*.

[24] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems* 2 (2020), 429–450.

[25] Xin-Chun Li, Yi-Chu Xu, Shaoming Song, Bingshuai Li, Yinchuan Li, Yunfeng Shao, and De-Chuan Zhan. [n. d.]. Federated Learning with Position-Aware Neurons. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. 10072–10081.

[26] Xin-Chun Li and De-Chuan Zhan. 2021. FedRS: Federated Learning with Restricted Softmax for Label Distribution Non-IID Data. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. 995–1005.

[27] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. [n. d.]. FedDG: Federated Domain Generalization on Medical Image Segmentation via Episodic Learning in Continuous Frequency Space. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. 1013–1023.

[28] Jianfeng Lu and Stefan Steinerberger. 2020. Neural Collapse with Cross-Entropy Loss. *CoRR* abs/2012.08465 (2020).

[29] Kangyang Luo, Xiang Li, Yunshi Lan, and Ming Gao. 2023. GradMA: A Gradient-Memory-based Accelerated Federated Learning with Alleviated Catastrophic Forgetting. *The IEEE/CVF Conference on Computer Vision and Pattern Recognition, (CVPR 2023), Vancouver, Canada* (2023).

[30] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. 2021. No Fear of Heterogeneity: Classifier Calibration for Federated Learning with Non-IID Data. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. 5972–5984.

[31] Grigory Malinovskiy, Dmitry Kovalev, Elnur Gasanov, Laurent Condat, and Peter Richtárik. 2020. From Local SGD to Local Fixed-Point Methods for Federated Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML, 13-18 July, Virtual Event*. 6692–6701.

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

[33] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*.

[34] Vardan Papyan. [n. d.]. Traces of Class/Cross-Class Structure Pervade Deep Learning Spectra. Journal of Machine Learning Research, 2020, 252(21):1–64. ([n. d.]).

[35] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated Learning for Emoji Prediction in a Mobile Keyboard. *CoRR*

abs/1906.04329 (2019).

[36] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. 2021. Adaptive Federated Optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7.*

[37] Yan Sun, Li Shen, Tiansheng Huang, Liang Ding, and Dacheng Tao. 2023. FedSpeed: Larger Local Interval, Less Communication Round, and Higher Generalization Accuracy. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5.*

[38] Chunnan Wang, Xiang Chen, Junzhe Wang, and Hongzhi Wang. June 18-24, 2022. ATPFL: Automatic Trajectory Prediction Model Design under Federated Learning Framework. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, New Orleans, LA, USA.* 6553–6562.

[39] Haozhao Wang, Yichen Li, Wenchao Xu, Ruixuan Li, Yufeng Zhan, and Zhigang Zeng. 2023. DaFKD: Domain-aware Federated Knowledge Distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023.* IEEE, 20412–20421.

[40] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. 2020. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

[41] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael G. Rabbat. 2020. SlowMo: Improving Communication-Efficient Distributed SGD with Slow Momentum. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.*

[42] Feijie Wu, Song Guo, Zhihao Qu, Shiqi He, Ziming Liu, and Jing Gao. 2023. Anchor Sampling for Federated Learning with Partial Client Participation. In *Proceedings of the 40th International Conference on Machine Learning.* 37379–37416.

[43] Feijie Wu, Song Guo, Haozhao Wang, Haobo Zhang, Zhihao Qu, Jie Zhang, and Ziming Liu. 2023. From Deterioration to Acceleration: A Calibration Approach to Rehabilitating Step Asynchronism in Federated Optimization. *IEEE Trans. Parallel Distributed Syst.* 34, 5 (2023), 1548–1559.

[44] An Xu, Wenqi Li, Pengfei Guo, Dong Yang, Holger Roth, Ali Hatamizadeh, Can Zhao, Daguang Xu, Heng Huang, and Ziyue Xu. [n. d.]. Closing the Generalization Gap of Cross-silo Federated Medical Image Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022.* 20834–20843.

[45] Jing Xu, Sen Wang, Liwei Wang, and Andrew Chi-Chih Yao. 2021. FedCM: Federated Learning with Client-level Momentum. *CoRR* abs/2106.10874 (2021).

[46] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian Nonparametric Federated Learning of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 7252–7261.

[47] Jie Zhang, Zhiqi Li, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Chao Wu. [n. d.]. Federated Learning with Label Distribution Skew via Logits Calibration. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA.* 26311–26329.

A Verification Experiment

To validate the motivations, we propose training a 2-layer MLP with 25 hidden neurons on the Mnist dataset, facilitating the visualization of neuron activations. In this experiment, each client is assigned a single data category to create a NonIID setting. Moreover, we demonstrate the change in activation states before and after local training on clients 1 and 2 by selecting a model from an intermediate training phase, where the chosen model has reached an accuracy of 88%. Below, we verify our viewpoints by comparing the activation values of the model's neurons on the data of clients 1 and 2, both before local training and post-local training; these activation values are presented in Table 4.

The existence of bounded neurons. Based on the original activation values on both clients, it can be seen that there is a significant difference in the bounded neurons between client1 and client2, thus validating our theory about bounded neurons.

The existence of neuron drift. It can be observed that the activation values of the model's bounded neurons on client2 decrease after local training on client1 using FedAvg.

Our method mitigates neuron drift. Based on the activation values after local training on client1 using `FedNLR`, it can be seen that the activation values of the bounded neurons also decrease on client2, while the extent of the decrease is smaller compared to FedAvg, thereby demonstrating that our method alleviates neuron drift.

B Proof of Theorem 3

B.1 Lemmas

Lemma 4. *We denote $\hat{\eta}^k$ by the learning rates for all parameters $\hat{\eta}^k = [\hat{\eta}^k_{1,1}, \ldots, \hat{\eta}^k_{1,M_1}, \ldots, \hat{\eta}^k_{L,M_L}]^T$. Let $\mu_{max} = \max(\mu_1, \ldots, \mu_L)$, $M_{min} = \min\{M_1, \ldots, M_L\}$, and $M_{max} = \max\{M_1, \ldots, M_L\}$. If $\hat{\eta}_{max}$ is the maximum learning rate for all parameters of all clients, i.e.,*

$$\hat{\eta}_{max} = \max(\max(\hat{\eta}^1), \ldots, \max(\hat{\eta}^K)),$$

then its upper bound is $\hat{\eta}_{max} \le \mu_{max}\eta/(\mu_{max} + M_{min} - 1)$. If $\hat{\eta}_{min}$ is the minimum learning rate for all parameters of all clients, i.e., $\hat{\eta}_{min} = \min(\min(\hat{\eta}^1), \ldots, \min(\hat{\eta}^K))$, then its lower bound is $\eta/(\mu_{max}(M_{max} - 1) + 1) \le \hat{\eta}_{min}$.

Proof: Based on the definition of learning rate $\hat{\eta}$ in (15), i.e.,

$$q_{l,m} = \frac{e^{\bar{h}_{l,m}/T_l}}{\sum_{i=1}^{M_l} e^{\bar{h}_{l,i}/T_l}}, \quad \hat{\eta}_{l,m} = \eta \cdot q_{l,m}, \tag{21}$$

we can directly obtain that $\sum_{m=1}^{M_l} \hat{\eta}_{l,m} = \eta$ for each layer l. Further, based on the definition of μ_l in (16), i.e., $\mu_l = \frac{\hat{\eta}_{l,\max}}{\hat{\eta}_{l,\min}}$, we can compute the upper bound of $\hat{\eta}_{l,\max}$ as

$$\hat{\eta}_{l,\max} \le \mu_l \eta/(\mu_l + M_l - 1) \tag{22}$$

when the learning rates of all $M_l - 1$ neurons are $\hat{\eta}_{l,\min}$. By denoting $\mu_{\max} = \max \mu_1, \ldots, \mu_L$ and $M_{\min} = \min M_1, \ldots, M_L$, we have the upper bound of the learning rate $\hat{\eta}_{l,\max}$ as

$$\hat{\eta}_{\max} \le \mu_{\max}\eta/(\mu_{\max} + M_{\min} - 1). \tag{23}$$

Similarly, we have the lower bound of the learning rate $\hat{\eta}_{l,\min}$ as

$$\eta/(\mu_{\max}(M_{\max} - 1) + 1) \le \hat{\eta}_{\min}, \tag{24}$$

which completes the proof.

Lemma 5. *The difference between the partially averaged gradient and the global gradient is bounded, i.e.,*

$$\mathbb{E}\left\| \frac{1}{Kr} \sum_{k \in \mathcal{K}_t} \hat{\eta}^k \odot \nabla F_k(\mathbf{w}^k_{t_*}) - \frac{1}{K} \sum_{k=1}^{K} \hat{\eta}^k \odot \nabla F_k(\mathbf{w}^k_{t_*}) \right\|_2^2 \le \hat{\eta}^2_{max} M^2 \frac{1-r}{r}.$$

Proof: We denote $\mathbf{1}(k)$ by the indicator function where $\mathbf{1}(k) = 1$ when the k-th client is selected and otherwise, $\mathbf{1}(k) = 0$. We have

$$\begin{aligned}
&\mathbb{E}\left\| \frac{1}{Kr} \sum_{k \in \mathcal{K}_t} \hat{\eta}^k \odot \nabla F_k(\mathbf{w}^k_{t_*}) - \frac{1}{K} \sum_{k=1}^{K} \hat{\eta}^k \odot \nabla F_k(\mathbf{w}^k_{t_*}) \right\|_2^2 \\
=&\mathbb{E}\left\| \frac{1}{Kr} \sum_{k=1}^{K} \mathbf{1}(k) \hat{\eta}^k \odot \nabla F_k(\mathbf{w}^k_{t_*}) - \frac{1}{K} \sum_{k=1}^{K} \hat{\eta}^k \odot \nabla F_k(\mathbf{w}^k_{t_*}) \right\|_2^2 \\
=&\mathbb{E}\left\| \frac{1}{K} \sum_{k=1}^{K} (1 - \frac{\mathbf{1}(k)}{r}) \hat{\eta}^k \odot \nabla F_k(\mathbf{w}^k_{t_*}) \right\|_2^2
\end{aligned}$$

Table 4: The activation of hidden neurons. The activation of bolded fonts represent bounded neurons. Activation change denotes the value change of bounded neurons on data of client 2 (C-2) between before and after training on client 1 (C-1).

NeuronID	1-8				9-16				17-25				
Original activation	0.39	0.16	0.12	0.24	0.17	**0.85**	0.1	0.49	0.02	0.32	0.34	0.02	0.07
on data of C-1	0.02	0	0	0.01	0	0.02	0.31	0.24	0.35	0.29	0.46	0.21	/
Original activation	0.07	0.33	0.04	0.06	0	0.1	0.23	0.14	**0.64**	0.22	0	0.2	0.07
on data of C-2	**0.56**	0	0	0.31	0	0.01	0.23	0.01	0	0.19	0.09	**0.52**	/
Activation on C-2	0.08	0.31	0.04	0.08	0	0.18	0.19	0.18	**0.61**	0.25	0	0.17	0.07
After C-1 via FedAvg	**0.50**	0	0	0.31	0	0.01	0.22	0.01	0.01	0.14	0.11	**0.46**	/
Activation Change	-0.03								-0.06			-0.06	
Activation on C-2	0.08	0.31	0.04	0.08	0	0.19	0.19	0.19	**0.63**	0.25	0	0.17	0.07
After C-1 via FedNLR	**0.53**	0	0	0.31	0	0.01	0.22	0.01	0.01	0.15	0.11	**0.48**	/
Activation Change	-0.01								-0.03			-0.04	

$$
\begin{aligned}
&\le \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}(1-\frac{\mathbb{1}(k)}{r})^2\left\|\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\right\|_2^2\\
&\le\frac{\hat{\eta}^2_{\max}}{K}\sum_{k=1}^{K}\mathbb{E}(1-\frac{\mathbb{1}(k)}{r})^2\|\nabla F_k(\mathbf{w}^k_{t_*})\|_2^2\\
&=\frac{\hat{\eta}^2_{\max}}{K}\sum_{k=1}^{K}\mathbb{E}\Big[r(1-\frac{1}{r})^2\|\nabla F_k(\mathbf{w}^k_{t_*})\|_2^2+(1-r)\|\nabla F_k(\mathbf{w}^k_{t_*})\|_2^2\Big]\\
&\le\hat{\eta}^2_{\max}M^2\frac{1-r}{r}, \qquad (25)
\end{aligned}
$$

which completes the proof.

B.2 Proof of Theorem 3

Without losing generality, we consider the data size $D_i = D_j$, for any two clients i and j. We denote the local iteration number e of the t-th round as $t_* = tE + e$. We also consider the number of participated clients $\mathcal{K}_t$ remains unchanged across different rounds with a ratio r from total K clients. Besides, we introduce another model for ease of analysis as

$$\mathbf{v}_{t_*} = \frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\mathbf{w}^k_{t_*} = \mathbf{v}_{t_*-1} - \frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla f_k(\mathbf{w}^k_{t_*-1}), \qquad (26)$$

where $\odot$ denotes the element-wise product. $\hat{\eta}^k$ denotes the vector of the learning rates for the parameters of all neurons $\hat{\eta}^k = [\hat{\eta}^k_{1,1},\dots,\hat{\eta}^k_{1,M_1},\dots,\hat{\eta}^k_{L,M_L}]^T$. Obviously, we have $\mathbf{v}_{t_*} = \mathbf{w}_{t_*}$ as $t_* \equiv 0 \pmod{E}$. Then, based on Assumption 1, we have

$$\mathbb{E}F(\mathbf{v}_{t_*+1}) \le \mathbb{E}F(\mathbf{v}_{t_*}) + \mathbb{E} < \nabla F_k(\mathbf{v}_{t_*}), \mathbf{v}_{t_*+1}-\mathbf{v}_{t_*} > + \frac{L}{2}\mathbb{E}\|\mathbf{v}_{t_*+1}-\mathbf{v}_{t_*}\|_2^2. \qquad (27)$$

According to the updated formula (26), we have

$$
\begin{aligned}
&\mathbb{E}\|\mathbf{v}_{t_*+1}-\mathbf{v}_{t_*}\|_2^2 = \mathbb{E}\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla f_k(\mathbf{w}^k_{t_*})\|_2^2\\
&=\mathbb{E}\Big\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla f_k(\mathbf{w}^k_{t_*}) - \frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\\
&\quad+\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2 \underset{(a)}{=} \mathbb{E}\Big\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2\\
&\quad+\mathbb{E}\Big\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla f_k(\mathbf{w}^k_{t_*}) - \frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2\\
&\underset{(b)}{=}\frac{1}{K^2r^2}\mathbb{E}\sum_{k\in\mathcal{K}_t}\Big\|\hat{\eta}^k\odot\big(\nabla f_k(\mathbf{w}^k_{t_*})-\nabla F_k(\mathbf{w}^k_{t_*})\big)\Big\|_2^2\\
&\quad+\mathbb{E}\Big\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2 \underset{(c)}{=} \mathbb{E}\Big\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2\\
&\quad+\frac{1}{K^2r}\mathbb{E}\sum_{k=1}^{K}\Big\|\hat{\eta}^k\odot\big(\nabla f_k(\mathbf{w}^k_{t_*})-\nabla F_k(\mathbf{w}^k_{t_*})\big)\Big\|_2^2\\
&\underset{(d)}{\le}\frac{\hat{\eta}^2_{\max}}{K^2r}\mathbb{E}\sum_{k=1}^{K}\Big\|\nabla f_k(\mathbf{w}^k_{t_*})-\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2+\mathbb{E}\Big\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2\\
&\underset{(e)}{\le}\frac{\hat{\eta}^2_{\max}\sigma_l^2}{Kr}+\mathbb{E}\Big\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2 \underset{(f)}{=} \mathbb{E}\Big\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2\\
&\quad+\frac{\hat{\eta}^2_{\max}\sigma_l^2}{Kr}+\mathbb{E}\Big\|\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})-\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2\\
&\underset{(g)}{\le}\frac{\hat{\eta}^2_{\max}\sigma_l^2}{Kr}+\hat{\eta}^2_{\max}M^2\frac{1-r}{r}+\mathbb{E}\Big\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^k\odot\nabla F_k(\mathbf{w}^k_{t_*})\Big\|_2^2. \qquad (28)
\end{aligned}
$$

where (a) and (f) holds because $\mathbb{E}\|a\|_2^2 = \mathbb{E}\|a-\mathbb{E}a\|_2^2 + \|\mathbb{E}a\|_2^2$. (b) is due to $\mathbb{E}f_k(\mathbf{w}^k_{t_*}) = \nabla F_k(\mathbf{w}^k_{t_*})$. (c) holds because $\mathbb{E}\sum_{k\in\mathcal{K}_t}a^k = r\mathbb{E}\sum_{k=1}^{K}a^k$. $\hat{\eta}_{\max}$ in the inequality (d) denotes the upper bound of the learning rate provided in Lemma 4. (e) is derived by Assumption 2. (g) is based on Lemma 5. We further note that

$$
\begin{aligned}
&\mathbb{E} < \nabla F(\mathbf{v}_{t_*}), \mathbf{v}_{t_*+1}-\mathbf{v}_{t_*} >\\
&=\mathbb{E} < \nabla F(\mathbf{v}_{t_*}), -\frac{1}{Kr}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla f_k(\mathbf{w}^k_{t_*}) >\\
&=-\mathbb{E}\sum_{l=1}^{L}\sum_{m=1}^{M_l}\frac{1}{\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^k_{l,m}} < \frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^k_{l,m}\nabla F(\mathbf{v}_{t_*})_{l,m},\\
&\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^k_{l,m}\nabla F_k(\mathbf{w}^k_{t_*})_{l,m} >= -\frac{1}{2}\mathbb{E}\sum_{l=1}^{L}\sum_{m=1}^{M_l}\frac{1}{\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^k_{l,m}}
\end{aligned}
$$

$$\Big(\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}\nabla F(\mathbf{v}_{t_*})_{l,m}\|_2^2+\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}\nabla F_k(\mathbf{w}_{t_*}^k)_{l,m}\|_2^2$$
$$-\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}\nabla F(\mathbf{v}_{t_*})_{l,m}-\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}\nabla F_k(\mathbf{w}_{t_*}^k)_{l,m}\|_2^2\Big)$$
$$=-\frac{1}{2K}\mathbb{E}\sum_{l=1}^{L}\sum_{m=1}^{M_l}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}\|\nabla F(\mathbf{v}_{t_*})_{l,m}\|_2^2$$
$$-\frac{1}{2}\mathbb{E}\sum_{l=1}^{L}\sum_{m=1}^{M_l}\frac{1}{\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}\nabla F_k(\mathbf{w}_{t_*}^k)_{l,m}\|_2^2$$
$$+\mathbb{E}\sum_{l=1}^{L}\sum_{m=1}^{M_l}\frac{1}{\frac{2}{K}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}_{l,m}^{k}\big(\nabla F(\mathbf{v}_{t_*})_{l,m}-\nabla F_k(\mathbf{w}_{t_*}^k)_{l,m}\big)\|_2^2$$
$$\le-\frac{\hat{\eta}_{\min}}{2K}\mathbb{E}\sum_{k=1}^{K}\|\nabla F(\mathbf{v}_{t_*})\|_2^2-\frac{1}{2\hat{\eta}_{\max}}\mathbb{E}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot\nabla F_k(\mathbf{w}_{t_*}^k)\|_2^2$$
$$+\frac{1}{2\hat{\eta}_{\min}}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot\big(\nabla F(\mathbf{v}_{t_*})-\nabla F_k(\mathbf{w}_{t_*}^k)\big)\|_2^2. \quad (29)$$

Bringing (29) and (29) back to (27), we have

$$\mathbb{E}F(\mathbf{v}_{t_*+1})\le\mathbb{E}F(\mathbf{v}_{t_*})-\frac{1}{2\hat{\eta}_{\max}}\mathbb{E}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot\nabla F_k(\mathbf{w}_{t_*}^k)\|_2^2$$
$$-\frac{\hat{\eta}_{\min}}{2K}\mathbb{E}\sum_{k=1}^{K}\|\nabla F(\mathbf{v}_{t_*})\|_2^2+\frac{1}{2\hat{\eta}_{\min}}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot\big(\nabla F(\mathbf{v}_{t_*})-\nabla F_k(\mathbf{w}_{t_*}^k)\big)\|_2^2$$
$$+\frac{\hat{\eta}_{\max}^2\sigma_l^2L}{2Kr}+\frac{(1-r)\hat{\eta}_{\max}^2M^2L}{2r}+\frac{L}{2}\mathbb{E}\left\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot\nabla F_k(\mathbf{w}_{t_*}^k)\right\|_2^2$$
$$\le\mathbb{E}F(\mathbf{v}_{t_*})+\big(\frac{L}{2}-\frac{1}{2\hat{\eta}_{\max}}\big)\mathbb{E}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot\nabla F_k(\mathbf{w}_{t_*}^k)\|_2^2$$
$$-\frac{\hat{\eta}_{\min}}{2K}\mathbb{E}\sum_{k=1}^{K}\|\nabla F(\mathbf{v}_{t_*})\|_2^2+\frac{\hat{\eta}_{\max}^2\sigma_l^2L}{2Kr}+\frac{(1-r)\hat{\eta}_{\max}^2M^2L}{2r}$$
$$+\frac{1}{2\hat{\eta}_{\min}}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot\big(\nabla F(\mathbf{v}_{t_*})-\nabla F_k(\mathbf{w}_{t_*}^k)\big)\|_2^2 \quad (30)$$
$$\le\mathbb{E}F(\mathbf{v}_{t_*})-\frac{\hat{\eta}_{\min}}{2K}\mathbb{E}\sum_{k=1}^{K}\|\nabla F(\mathbf{v}_{t_*})\|_2^2+\frac{\hat{\eta}_{\max}^2\sigma_l^2L}{2Kr}+\frac{(1-r)\hat{\eta}_{\max}^2M^2L}{2r}$$
$$+\frac{1}{2\hat{\eta}_{\min}}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot\big(\nabla F(\mathbf{v}_{t_*})-\nabla F_k(\mathbf{w}_{t_*}^k)\big)\|_2^2\le\mathbb{E}F(\mathbf{v}_{t_*})$$
$$-\frac{\hat{\eta}_{\min}}{2K}\mathbb{E}\sum_{k=1}^{K}\|\nabla F(\mathbf{v}_{t_*})\|_2^2+\frac{2(\hat{\eta}_{\max}^k)^3E^2M^2}{\hat{\eta}_{\min}}+\frac{\hat{\eta}_{\max}^2\sigma_l^2L}{2Kr}+\frac{(1-r)\hat{\eta}_{\max}^2M^2L}{2r},$$

where the last-second inequality holds when $\frac{L}{2}-\frac{1}{2\hat{\eta}_{\max}}<0$, i.e., $\hat{\eta}_{\max}<\frac{1}{L}$. The last inequality holds because

$$\mathbb{E}\|\frac{1}{K}\sum_{k=1}^{K}\hat{\eta}^{k}\odot(\nabla F(\mathbf{v}_{t_*})-\nabla F_k(\mathbf{w}_{t_*}^k))\|_2^2$$
$$\le\frac{\hat{\eta}_{\max}^2}{K}\sum_{k=1}^{K}\mathbb{E}\|\nabla F_k(\mathbf{v}_{t_*})-\nabla F_k(\mathbf{w}_{t_*}^k)\|_2^2\le\frac{\hat{\eta}_{\max}^2L^2}{K}\sum_{k=1}^{K}\mathbb{E}\|\mathbf{v}_{t_*}-\mathbf{w}_{t_*}^k\|_2^2$$
$$=\frac{\hat{\eta}_{\max}^2L^2}{K}\sum_{k=1}^{K}\mathbb{E}\left\|\frac{1}{Kr}\sum_{\tau=0}^{e}\sum_{k\in\mathcal{K}_t}\hat{\eta}^k\odot\nabla f_k(\mathbf{w}_{t_*-e+\tau}^k)-\sum_{\tau=0}^{e}\hat{\eta}^k\odot\nabla f_k(\mathbf{w}_{t_*-e+\tau}^k)\right\|_2^2$$
$$\le\frac{2\hat{\eta}_{\max}^2L^2e}{Kr}\sum_{k=1}^{K}\sum_{\tau=0}^{e}\mathbb{E}\sum_{k\in\mathcal{K}_t}\|\hat{\eta}^k\odot\nabla f_k(\mathbf{w}_{t_*-e+\tau}^k)\|_2^2$$
$$+\frac{2\hat{\eta}_{\max}^2L^2e}{K}\sum_{k=1}^{K}\sum_{\tau=0}^{e}\mathbb{E}\|\hat{\eta}^k\odot\nabla f_k(\mathbf{w}_{t_*-e+\tau}^k)\|_2^2$$
$$\le\frac{2(\hat{\eta}_{\max})^3e}{K}\sum_{\tau=0}^{e}\sum_{k=1}^{K}\mathbb{E}\|\nabla f_k(\mathbf{w}_{t_*-e+\tau}^k)\|_2^2$$
$$+\frac{2(\hat{\eta}_{\max})^3e}{K}\sum_{\tau=0}^{e}\sum_{k=1}^{K}\mathbb{E}\|\nabla f_k(\mathbf{w}_{t_*-e+\tau}^k)\|_2^2\le4(\hat{\eta}_{\max}^k)^3E^2M^2. \quad (31)$$

Since $\hat{\eta}_{\min}\|\nabla F(\mathbf{v}_{t_*})\|_2^2\le\|\sqrt{\hat{\eta}^k}\odot\nabla F_k(\mathbf{v}_{t_*})\|_2^2$, re-arranging (30) obtains

$$\frac{\hat{\eta}_{\min}}{2K}\mathbb{E}\sum_{k=1}^{K}\|\nabla F(\mathbf{v}_{t_*})\|_2^2\le\mathbb{E}F(\mathbf{v}_{t_*})-\mathbb{E}F(\mathbf{v}_{t_*+1})$$
$$+\frac{2(\hat{\eta}_{\max}^k)^3E^2M^2}{\hat{\eta}_{\min}}+\frac{\hat{\eta}_{\max}^2\sigma_l^2L}{2Kr}+\frac{(1-r)\hat{\eta}_{\max}^2M^2L}{2r}. \quad (32)$$

By taking a sum of both sides of (32) from $t_*=1$ to T_*, we have

$$\sum_{t_*=1}^{T_*}\mathbb{E}\|\nabla F(\mathbf{v}_{t_*})\|_2^2\le\frac{2(\mathbb{E}F(\mathbf{v}_1)-\mathbb{E}F(\mathbf{v}_{T_*}))}{\hat{\eta}_{\min}}$$
$$+\frac{4(\hat{\eta}_{\max}^k)^3E^2M^2T}{\hat{\eta}_{\min}^2}+\frac{\hat{\eta}_{\max}^2\sigma_l^2LT}{\hat{\eta}_{\min}Kr}+\frac{(1-r)\hat{\eta}_{\max}^2M^2LT}{\hat{\eta}_{\min}r}. \quad (33)$$

Denote F_* as the global optima. Since $\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|_2^2\le\sum_{t_*=1}^{T_*}\mathbb{E}\|\nabla F(\mathbf{v}_{t_*})\|_2^2$, by setting $\eta=\sqrt{\frac{K}{T}}$, based on Lemma 4, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\mathbf{w}_t)\|_2^2\le\frac{2(F(\mathbf{w}_1)-F_*))}{\hat{\eta}_{\min}T}+\frac{4(\hat{\eta}_{\max}^k)^3E^2M^2}{\hat{\eta}_{\min}^2}$$
$$+\frac{\hat{\eta}_{\max}^2\sigma_l^2L}{\hat{\eta}_{\min}Kr}+\frac{(1-r)\hat{\eta}_{\max}^2M^2L}{\hat{\eta}_{\min}r}$$
$$\le\frac{2(F(\mathbf{w}_1)-F_*))\sqrt{\mu_{\max}(M_{\max}-1)+1}}{\sqrt{KT}}$$
$$+\frac{4\mu_{\max}^3(\mu_{\max}(M_{\max}-1)+1)^2E^2M^2}{(\mu_{\max}+M_{\min}-1)^3\sqrt{T}}$$
$$+\frac{\mu_{\max}^2(\mu_{\max}(M_{\max}-1)+1)\sigma_l^2L}{(\mu_{\max}+M_{\min}-1)^2r\sqrt{KT}}$$
$$+\frac{(1-r)\mu_{\max}^2(\mu_{\max}(M_{\max}-1)+1)M^2L}{(\mu_{\max}+M_{\min}-1)^2r\sqrt{T}}, \quad (34)$$

which completes the proof.