Technical Appendix

The Pseudocode of Our Method

Algorithm 1: **FedVLS**

Input: number of communication rounds T, number of clients N, client participating rate R, number of local epochs E, batch size B, learning rate η.
Output: the global model $\boldsymbol{\omega}^T$

initialize $\boldsymbol{\omega}^0$
$m \leftarrow \max(\lfloor R \cdot N \rfloor, 1)$
for *communication round* $t = 1, 2, \cdots, T-1$ **do**
  $M_t \leftarrow$ randomly select a subset containing m clients
  for *each client* $i \in M_t$ **do**
    $\boldsymbol{\omega}_i^t = \boldsymbol{\omega}^t$
    $\boldsymbol{\omega}_i^{t+1} \leftarrow$ **LocalUpdate**$(\boldsymbol{\omega}_i^t)$
  end for
  $\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t + \sum_{i \in M_t} \frac{|\mathcal{D}_i|}{|\mathcal{D}|}(\boldsymbol{\omega}_i^{t+1} - \boldsymbol{\omega}_i^t)$
end for

LocalUpdate$(\boldsymbol{\omega}_i^t)$:
for *epoch* $e = 1, 2, \cdots, E$ **do**
  for *each batch* $\mathcal{B}_i = \{x, y\} \in \mathcal{D}_i$ **do**
    $\mathcal{L}_{\mathbf{cal}}(\boldsymbol{\omega}; \mathcal{B}_i) = -\mathbb{E}_{(x,y)\sim\mathcal{B}_i} \log\left(\frac{p(y)\cdot \mathbf{e}^{f(x;\boldsymbol{\omega})[y]}}{\sum_c p(c)\cdot \mathbf{e}^{f(x;\boldsymbol{\omega})[c]}}\right)$
    $\mathcal{L}_{\mathbf{dis}}(\boldsymbol{\omega}; \mathcal{B}_i) = \mathbb{E}_{(x,y)\sim\mathcal{B}_i} \sum_{o\in\mathbb{O}} q^{\mathbf{g}}(o;x) \log\left[\frac{q(o;x)}{q^{\mathbf{g}}(o;x)}\right]$
    $\mathcal{L}^c_{\mathbf{logit}}(\boldsymbol{\omega}; \mathcal{B}_i) = \log\left(\mathbb{E}_{(x,y)\sim\mathcal{B}_i} \mathbb{I}(y \neq c) \cdot \mathbf{e}^{f(x;\boldsymbol{\omega})[c]}\right)$
    $\mathcal{L}_{\mathbf{logit}}(\boldsymbol{\omega}; \mathcal{B}_i) = \sum p(c) \cdot \mathcal{L}^c_{\mathbf{logit}}(\boldsymbol{\omega}; \mathcal{B}_i)$
    $\mathcal{L}(\boldsymbol{\omega}_i^t; \mathcal{B}_i) = \mathcal{L}_{\mathbf{cal}}(\boldsymbol{\omega}_i^t; \mathcal{B}_i) + \lambda \cdot \mathcal{L}_{\mathbf{dis}}(\boldsymbol{\omega}_i^t; \mathcal{B}_i) + \mathcal{L}_{\mathbf{logit}}(\boldsymbol{\omega}_i^t; \mathcal{B}_i)$
    $\boldsymbol{\omega}_i^t = \boldsymbol{\omega}_i^t - \eta \nabla \mathcal{L}(\boldsymbol{\omega}_i^t; \mathcal{B}_i)$
  end for
end for
return $\boldsymbol{\omega}_i^t$

Experimental Details

Data Distribution among Clients

In Figure 1 (a) of the main paper, all clients' data distributions are independent and identically sampled. In Figures1 (b), (c), (d) of the main paper, the data distribution of all clients is shown in Table 7 as follows. We focus on client 0 for analysis, where it is evident that classes 5, 8, and 9 are majority classes, class 3 is a minority class, and the remaining classes are vacant.

In Figure 2 of the main paper, the data distribution for this client is shown in the fourth column of Figure 7 (a). Here, classes 0, 1, 3, and 7 are majority classes, while classes 2, 5, and 6 are minority classes. Figure 6 reveals that minority classes are frequently misclassified as majority classes, which motivates the introduction of Logit Suppression in the main paper.

In our experiments, we incorporate Dirichlet-based label skews ($\beta = 0.5, 0.1, 0.05$) and quantity-based label skews (s=2) for the CIFAR10 dataset. The data distribution for these skews is illustrated in Figure 7.

Table 7: The data distribution among clients with Dirichlet-based ($\beta = 0.1$) CIFAR10 datasets.

client	0	1	2	3	4	5	6	7	8	9
class 0	0	57	0	600	0	4342	0	0	0	1
class 1	0	155	0	0	1	679	4153	0	11	1
class 2	0	3	24	0	15	0	3536	1419	0	3
class 3	141	99	3490	953	0	0	0	0	208	109
class 4	0	0	98	1217	3684	0	0	0	1	0
class 5	1471	0	3403	0	125	0	0	0	0	1
class 6	0	0	0	0	0	0	0	4999	1	0
class 7	0	0	0	2	0	0	0	0	4998	0
class 8	1360	35	0	0	3604	0	0	0	0	1
class 9	366	4608	0	0	0	0	0	0	0	26

Implementation Details

The augmentation for all CIFAR and TinyImageNet experiments is the same as existing literature AutoAugment (Cubuk et al. 2019). The specific architecture of MobileNetV2 (Sandler et al. 2018) is shown in Table 8, while the structure of the bottleneck is detailed in Table 9. Since the architectures of ResNet-18 and ResNet-32 are well-known, we do not present their detailed structures here. Hyperparameters for all baseline methods are set according to the configurations specified in the original papers, as detailed in Table 10. All experiments are conducted on a single NVIDIA GeForce RTX 3090 with 24GB of memory.

Table 8: The architecture of MobileNetV2.

Input	Operator	t	*C*	*n*	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d1 $\times$ 1	-	1280	1	1
$7^2 \times 1280$	avgpool7 $\times$ 7	-	-	1	-
$1 \times 1 \times 1280$	conv2d1 $\times$ 1	-	k	-	

Additional Experimental Observations

In Figure 5, the updated local model's performance on classes 5 and 8 surpasses that of the initial global model.

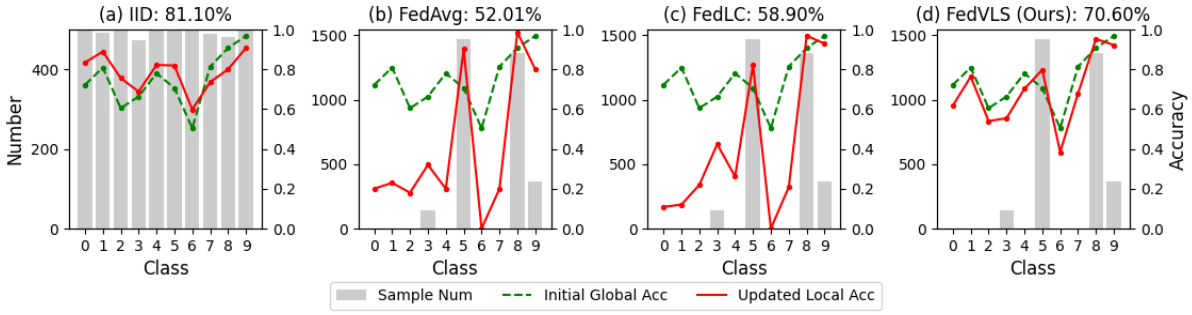

Figure 5: Class-wise accuracy of the initial global model and updated local model on IID and label-skewed CIFAR10 data distributions. (a) represents the result updating on IID local data with FedAvg (McMahan et al. 2017). (b-d) showcase the results updating on skewed local data distribution with FedAvg, FedLC (Zhang et al. 2022), and our FedVLS, respectively. The value (%) in each caption corresponds to the accuracy of the global model aggregated from updated local models.

Figure 6: Confusion matrix of client 3 on CIFAR10 dataset with Dirichlet-based label skews (β = 0.5) using FedLC (Zhang et al. 2022).

Table 9: The architecture of bottleneck.

Input	Operator	Output
$h \times w \times k$	1×1 conv2d, ReLU6	$h \times w \times (tk)$
n$h \times w \times tk$	3×3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$ $\frac{h}{s} \times \frac{w}{s} \times k'$
n$\frac{h}{s} \times \frac{w}{s} \times tk$	linear1 $\times$ 1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times (tk)$ $\frac{h}{s} \times \frac{w}{s} \times k'$

Table 10: The hyperparameters for all baseline methods.

FedAvg (AISTATS 2017)	None
FedProx (MLSys 2020)	μ=0.01
MOON (CVPR 2021)	μ=0.01, τ=0.5
FedEXP (ICLR 2023)	ϵ=0.01
FedLC (ICML 2022)	τ=0.5
FedRS (KDD 2021)	α=0.7
FedSAM (ICML2022)	ρ=0.1, β=0.9
FedNTD (NeurIPS 2022)	β=0.1
FedMR (TMLR 2023)	$deco$=4
FedLMD (MM 2023)	β=0.1
FedConcat (AAAI 2024)	$cluster$={2, 4}
FedGF (ICML 2024)	ρ=0.1, c_os=0.3

This improvement is due to our proposed loss function, which constrains the local model's output for vacant classes and suppresses the misclassification of minority samples. These adjustments have minimal impact on the learning of majority classes. Consequently, local models continue to acquire category knowledge from majority classes, such as classes 5 and 8, similar to FedAvg, resulting in enhanced classification accuracy for these classes.

Another interesting observation is that both FedLC (Zhang et al. 2022) and our method reduce the accuracy of classes 5 and 8 while increasing the accuracy of the remaining classes. The reason for this behavior is as follows: In FedAvg (McMahan et al. 2017), the local model often misclassifies vacant and minority classes as majority classes. This leads to disproportionately high accuracy for the majority classes and extremely low accuracy for the minority and vacant classes.

FedLC (Zhang et al. 2022) employs logit weighting to enhance the learning of minority classes, which can result in some majority class samples being misclassified as similar minority classes. As a result, this method improves accuracy for minority classes while slightly reducing accuracy for majority classes. In contrast, our method introduces vacant-class distillation and logit suppression to substantially mitigate the misclassification of minority and vacant classes as majority classes. This approach improves accuracy for vacant and minority classes but may cause some majority class samples to be misclassified as similar vacant or minority classes. Consequently, while this slightly reduces accuracy for the majority classes, it significantly enhances the overall performance of the local models.

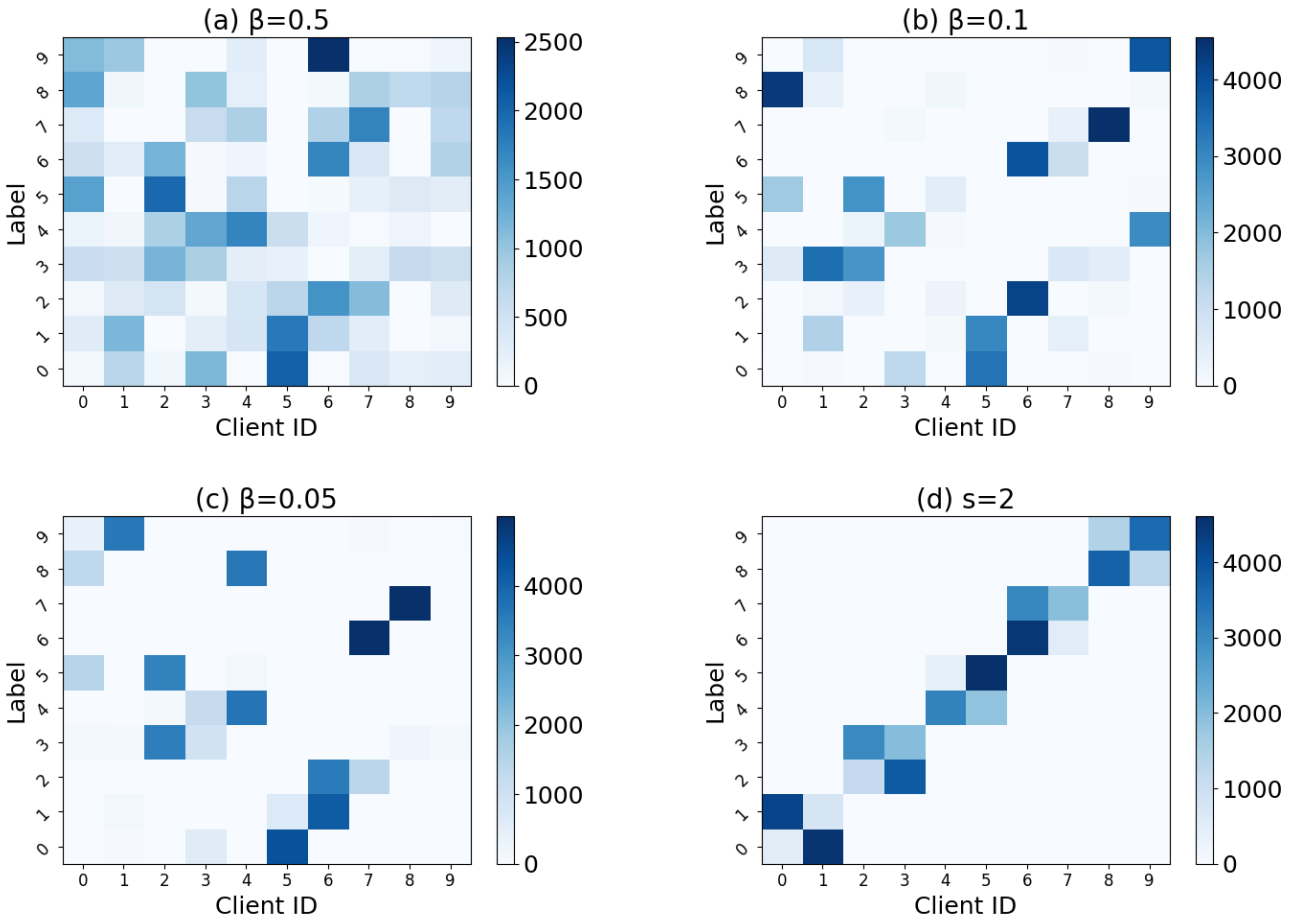

Figure 7: Visualization of the Dirichlet-based ($\beta = 0.5, 0.1, 0.05$) and quantity-based (s=2) label skews of CIFAR10 dataset among 10 clients.

Additional Experimental Results

The Experimental Results on the AG_news Dataset

In this subsection, we add the experimental results on the AG_news dataset with Dirichlet-based ($\beta = 0.1$ and $\beta = 0.05$) and quantity-based (s=2) label skews, as shown in the Tab 11, demonstrating our method, FedVLS, consistently outperforms the base- line methods. These experiments underscore the versatility and robustness of FedVLS in real-world federated learning scenarios facing text classification.

Compared to Other Knowledge Distillation Methods

To demonstrate the effectiveness of our vacant-class distillation, we compare it with existing class distillation, normal distillation, DKD (Zhao et al. 2022), and FedNTD (Lee et al. 2022). Similar to FedNTD (Lee et al. 2022), we integrate existing class distillation, normal distillation (KD), and DKD (Zhao et al. 2022) into FedAvg, denoted as FedEKD, FedKD, and FedDKD, respectively. As shown in Table 13, our method consistently outperforms these approaches.

Table 11: Performance overview for our method and baselines on the **AG_news** dataset with Dirichlet-based (β=0.05 and β=0.1) and quantity-based (s=2) label skews. **Bold** is the best result.

Method(venue)	$\beta = 0.1$	$\beta = 0.05$	$s = 2$
FedAvg (AISTATS 2017)	73.52	71.08	62.85
FedProx (MLSys 2020)	75.11	71.92	64.36
FedEXP (ICLR 2023)	78.08	72.35	63.01
FedSAM (ICML2022)	77.88	72.46	66.73
FedNTD (NeurIPS 2022)	79.14	75.60	69.28
FedLMD (MM 2023)	82.14	77.54	71.41
FedConcat (AAAI 2024)	81.59	74.84	68.11
FedGF (ICML 2024)	82.76	77.09	70.28
FedVLS (Ours)	**87.31**	**83.19**	**77.46**

To investigate the underlying reasons, we further examined the class-wise accuracy of the initial global model and the local models trained using these methods on client 0, whose data distribution is detailed in Table 7. The specific class-wise accuracy results are presented in Table 12. FedEKD shows minimal improvement in majority classes

Table 12: The class-wise accuracy for different knowledge distillation methods with Dirichlet-based ($\beta = 0.1$) CIFAR10 datasets.

class	1	2	3	4	5	6	7	8	9	10	Avg
global model	72.20	90.90	71.30	72.10	84.40	73.60	86.20	73.80	90.60	93.80	80.89
FedAvg	0	0	0	44.10	0	98.40	0	0	97.90	95.90	33.63
FedEKD	0	0	0	45.30	0	**98.80**	0	0	**98.90**	94.90	33.79
FedKD	1.50	5.10	1.80	51.90	1.60	94.80	1.20	0	97.40	95.50	35.08
FedDKD	3.90	34.20	16.60	52.50	9.80	94.10	14.40	0.10	97.60	**96.00**	41.92
FedNTD	8.00	38.80	22.60	58.10	11.10	96.10	12.20	0.20	98.10	94.60	43.98
Ours	**40.40**	**71.20**	**39.60**	**64.60**	**50.07**	83.50	**54.80**	**41.30**	92.30	94.32	63.21

Table 13: Performance overview for different knowledge distillation methods under Dirichlet-based label skews.

Method	CIFAR10		CIFAR100		TinyImageNet	
	$\beta = 0.1$	$\beta = 0.05$	$\beta = 0.1$	$\beta = 0.05$	$\beta = 0.1$	$\beta = 0.05$
FedAvg	82.00	62.90	66.18	62.13	39.90	35.21
FedEKD	81.25	62.26	67.66	62.90	40.95	36.13
FedKD	82.42	64.16	67.19	63.21	41.77	36.55
FedDKD	82.87	65.27	67.70	63.53	43.63	37.23
FedNTD	83.23	68.71	68.00	63.71	45.11	40.65
Ours	**84.35**	**75.71**	**69.02**	**65.71**	**47.73**	**45.23**

but significantly hinders the learning of vacant classes. FedKD, which uses distillation across all classes, still exhibits low accuracy for vacant classes. FedDKD adjusts distillation weights for true and not-true classes, while FedNTD applies distillation to not-true classes. Although these methods improve accuracy for vacant classes, there remains a substantial gap compared to the global model. Based on these observations, we believe that performing distillation on majority and minority classes will weaken the protection of information for vacant classes. Therefore, we use vacant-class distillation. The results in Table 12 further demonstrate that our method significantly enhances the accuracy for vacant classes, finally improving the performance of the local and global models.

Combined with Methods for Domain Shift

Our method is specifically designed to address label skews, making it complementary to approaches that tackle domain skews. When both domain and label skews are present, our approach can further enhance the performance of methods like FPL (Huang et al. 2023). We have conducted experiments to validate this, with results presented in Table 14 and Table 15. Following the experimental setup in FPL (Huang et al. 2023), we use the Digits dataset and apply Dirichlet sampling to distribute the data for each domain among six clients. Under conditions of both domain and label skews, our method significantly improves the performance of PFL (Huang et al. 2023), demonstrating its effectiveness across different levels of label skews and domain shifts.

Table 14: Performance overview for FPL and our method combined with FPL in Dirichlet-based label skews, β=0.1. **Bold** is the best result.

Method	MNIST	USPS	SVHN	SYN	AVG
FPL	97.56	**98.73**	85.06	94.23	93.89
FPL + Ours	**98.36**	98.40	**86.66**	**95.38**	**94.70**

Table 15: Performance overview for FPL and our method combined with FPL in Dirichlet-based label skews, β=0.05. **Bold** is the best result.

Method	MNIST	USPS	SVHN	SYN	AVG
FPL	96.82	96.40	77.09	89.96	90.07
FPL + Ours	**97.75**	**97.07**	**82.05**	**91.74**	**92.15**

Impact of Communication Rounds

In real-world scenarios, constraints often limit the number of available communication rounds. To address this, we evaluate the performance of various methods under different communication round limits using the CIFAR10 dataset with skew parameters $\beta = 0.1$ and $\beta = 0.05$. The results, presented in Table 16, show that as the number of communication rounds decreases, the accuracy of most methods drops significantly. However, our method maintains high accuracy even with fewer communication rounds, demonstrating the robustness and efficiency of FedVLS in environments with restricted communication capabilities.

Impact of Joining Rates, Local Epochs, and Client Numbers

Due to space constraints, we included only a portion of the ablation studies on joining rates, local epochs, and client numbers in the main paper. Here, we present the complete results. Specifically, we evaluated joining rates of 0.3, 0.5, 0.8, 1.0, local epochs of 5, 10, 15, 20, and client numbers of 10, 20, 30, 50. The experimental results are shown in Figure 8, and the observations are consistent with those presented in the main paper.

As the participation rate decreases, several methods exhibit highly unstable convergence. In contrast, our method

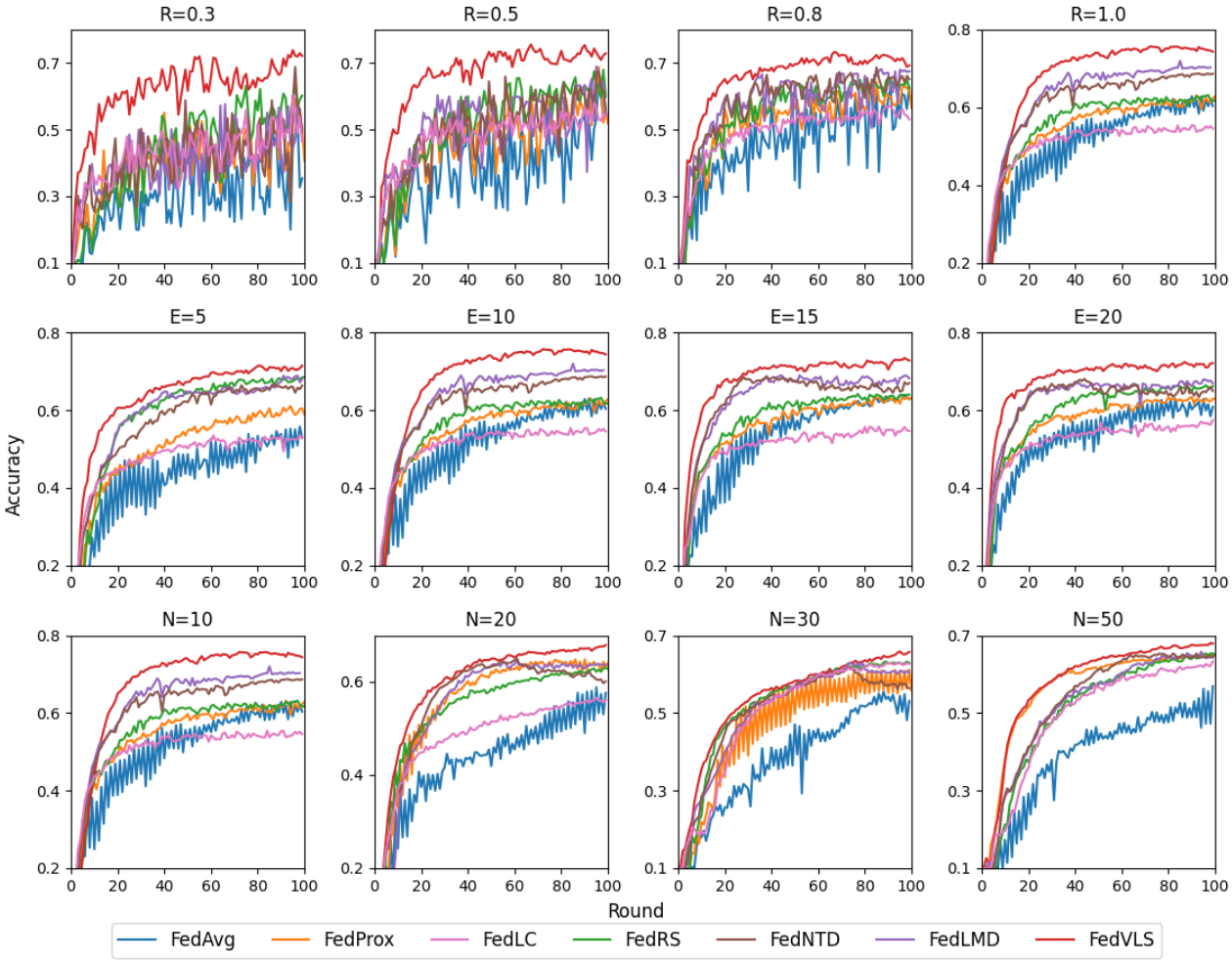

Figure 8: Sensitivity analysis on the client participating rate $\mathbf{R}$, local epochs $\mathbf{E}$, and client numbers $\mathbf{N}$. Each figure separately shows the convergence curve with Dirichlet-based label skews ($\beta = 0.05$) on CIFAR10 dataset with $\mathbf{R}$ in $\{0.3, 0.5, 0.8, 1.0\}$, $\mathbf{E}$ in $\{5, 10, 15, 20\}$ and $\mathbf{N}$ in $\{10, 20, 30, 50\}$.

demonstrates relatively stable convergence, highlighting its robustness to varying participation rates.

Increasing the number of local epochs leads to declining accuracy in the later stages of training for several methods, notably FedNTD (Lee et al. 2022). However, our method maintains consistency and improves performance with larger E values, consistently outperforming other methods.

With an increasing number of clients, many methods show slower and less stable convergence. This is because the larger the number of clients, the greater the damage to model convergence caused by data heterogeneity among clients. However, our method maintains rapid and stable convergence across varying client numbers, demonstrating the robustness and scalability of our approach.

Class-wise Accuracy

To evaluate the effectiveness of our approach, we conduct a comparative analysis of class-wise accuracy before and after local updates using our method, the classic method FedAvg (McMahan et al. 2017), and the state-of-the-art method FedLC (Zhang et al. 2022). For a fair comparison, we use the same well-trained federated model as the initial global model, which is then distributed to all clients. We train the local models using FedAvg and FedLC, and our method uses the same local data distribution. As shown in Figure 1 of the main paper, the results align with the observations discussed in the motivation section. Additionally, we compare the average class-wise accuracy for all clients after local updates and the class-wise accuracy for the aggregated global model of our approach with that of FedLC (Zhang et al. 2022), as demonstrated in Figure 9. Our method consistently achieves higher class-wise accuracy compared to FedLC, both after local updates and model aggregation.

Table 16: Results under varying numbers of communication rounds with Dirichlet-based label skews on CIFAR10 dataset.

Method(venue)	40 comm		60 comm		80 comm	
	β=0.1	β=0.05	β=0.1	β=0.05	β=0.1	β=0.05
FedAvg (AISTATS 2017)	74.62	53.44	78.59	56.71	80.72	59.10
FedProx (MLSys 2020)	78.59	57.67	81.63	61.84	82.88	61.96
MOON (CVPR 2021)	78.23	52.84	81.73	57.11	82.91	61.35
FedEXP (ICLR 2023)	75.90	54.14	79.69	55.98	81.51	60.01
FedLC (ICML 2022)	75.74	53.06	77.22	53.77	80.22	55.75
FedRS (KDD 2021)	79.10	60.99	81.13	63.16	82.94	64.28
FedSAM (ICML2022)	69.02	50.05	75.42	55.85	78.38	60.79
FedNTD (NeurIPS 2022)	81.26	65.75	82.23	66.48	82.95	67.91
FedLMD (MM 2023)	79.99	66.72	81.77	68.14	83.01	69.87
FedVLS (Ours)	**82.54**	**72.90**	**83.82**	**74.34**	**84.30**	**75.25**

These results highlight how our method effectively improves the performance of minority and vacant classes, leading to an overall enhancement in the global model's performance.

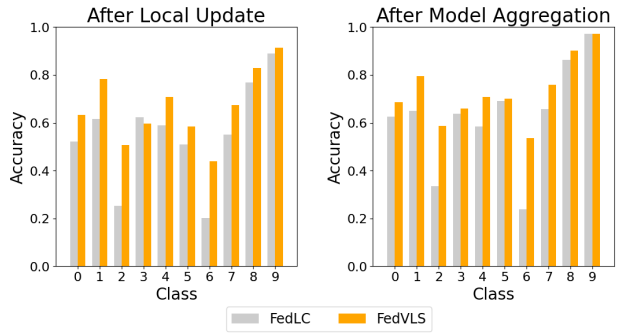

Figure 9: Comparison of class-wise accuracy after local update and after model aggregation with Dirichlet-based label skews ($\beta = 0.05$) on CIFAR10 dataset.

Model Bias among Clients

Thanks to the Vacant-class Distillation module, the client model will pay more attention to the vacant classes, which is beneficial to alleviate the model bias among clients. To demonstrate this, we conduct experiments to measure the drift diversity across all client models in the final round following (Li et al. 2023). Specially, the drift diversity is defined as follows:

$$Drift = \frac{\sum_{i=1}^{N} \|m_i\|^2}{\|\sum_{i=1}^{N} m_i\|^2}, m_i = \omega_i^T - \omega^T \tag{7}$$

The results are presented in Table 17. It is evident that our approach effectively mitigates model bias among clients, leading to improved global performance.

The Connection between Equation (2) of The Main Paper and FedLC

Apart from FedLC (Zhang et al. 2022), Fedshift (Shen, Wang, and Lv 2023) also adjusts the logits of model outputs to alleviate model bias caused by imbalanced data distributions. However, they have different forms, so we uniformly

Table 17: The drift diversity of different method on CIFAR10 datasets with $\beta = 0.1$.

Method	FedAvg	FedNTD	FedLC	FedVLS (Ours)
Drift diversity	29.73	17.85	12.11	**8.37**

represent their loss functions using Eq(2). Nevertheless, during experiments, we train the models according to the original loss function forms as presented in the respective papers. Below, we demonstrate that Eq(2) is positively correlated to the loss function in FedLC (Zhang et al. 2022). In Eq(2),

$$\mathcal{L}_{\mathbf{cal}} = -\mathbb{E}_{(x,y)\sim\mathcal{D}_i} \log \left(\frac{p(y) \cdot e^{f(x;\boldsymbol{\omega})[y]}}{\sum_c p(c) \cdot e^{f(x;\boldsymbol{\omega})[c]}} \right) \tag{8}$$

$$= -\mathbb{E}_{(x,y)\sim\mathcal{D}_i} \log \left(\frac{e^{\ln p(y)} \cdot e^{f(x;\boldsymbol{\omega})[y]}}{\sum_c e^{\ln p(c)} \cdot e^{f(x;\boldsymbol{\omega})[c]}} \right) \tag{9}$$

$$= -\mathbb{E}_{(x,y)\sim\mathcal{D}_i} \log \left(\frac{e^{\ln p(y) + f(x;\boldsymbol{\omega})[y]}}{\sum_c e^{\ln p(c) + f(x;\boldsymbol{\omega})[c]}} \right), \tag{10}$$

where $p(y) = \frac{n_y}{n}$, n_y is the number of samples of class y in client i, and n is the total number of samples in client i. Therefore, Eq(2) can be rewritten in the following form.

$$\mathcal{L}_{\mathbf{cal}} = -\mathbb{E}_{(x,y)\sim\mathcal{D}_i} \log \left(\frac{e^{\ln\left(\frac{n_y}{n}\right) + f(x;\boldsymbol{\omega})[y]}}{\sum_c e^{\ln\left(\frac{n_c}{n}\right) + f(x;\boldsymbol{\omega})[c]}} \right) \tag{11}$$

$$= -\mathbb{E}_{(x,y)\sim\mathcal{D}_i} \log \left(\frac{e^{f(x;\boldsymbol{\omega})[y] + \ln n_y - \ln n}}{\sum_c e^{f(x;\boldsymbol{\omega})[c] + \ln n_c + \ln n}} \right) \tag{12}$$

For different classes within the same client, n remains the same while n_y varies. Therefore, the loss functions for different classes lie in n_y and the output logits. Compared with the loss function in FedLC,

$$\mathcal{L}_{\mathbf{cal}}(y, f(x)) = -\log \left(\frac{e^{f_y(x) - \tau \cdot n_y^{(-1/4)}}}{\sum_{c \neq y} e^{f_c(x) - \tau \cdot n_y^{(-1/4)}}} \right), \tag{13}$$

$\ln n_y$ and $-\tau \cdot n_y^{(-1/4)}$ exhibit the same trend as n_y changes, therefore they have similar effects on the loss function.