Mask-Encoded Sparsification: Mitigating Biased Gradients in Communication-Efficient Split Learning

Paper #2121

Abstract. This paper introduces a novel framework designed to achieve a high compression ratio in Split Learning (SL) scenarios where resource-constrained devices are involved in large-scale model training. Our investigations demonstrate that compressing feature maps within SL leads to biased gradients that can negatively impact the convergence rates and diminish the generalization capabilities of the resulting models. Our theoretical analysis provides insights into how compression errors critically hinder SL performance, which previous methodologies underestimate. To address these challenges, we employ a narrow bit-width encoded mask to compensate for the sparsification error without increasing the order of time complexity. Supported by rigorous theoretical analysis, our framework significantly reduces compression errors and accelerates the convergence. Extensive experiments also verify that our method outperforms existing solutions regarding training efficiency and communication complexity.

1 Introduction

In recent years, there has been an upsurge in demand for training deep neural networks (DNNs) [9, 11, 29, 39] on limited-resource end devices such as smartphones and embedded AI chips, primarily for edge intelligence applications such as speech recognition and intelligent healthcare. Federated learning (FL) [25, 34], a widely-used distributed learning approach, allows a group of end devices to train a global model collaboratively without sharing personal data. However, FL requires training the entire DNN on these local devices before aggregating the results on a central server, which can lead to server underutilization and strain on these devices' computational and energy resources. Moreover, as DNNs become larger and more complex [21, 23, 29], this issue becomes more acute and potentially limits the feasibility of FL in resource-constrained environments.

To solve this problem, Split Learning (SL) [14, 16, 18, 43, 46] is increasingly emerging as a promising approach that leverages the capabilities of both the server and end devices. Specifically, SL splits the DNN models into two parts, namely the client-side model and the server-side model, which are deployed on multiple end devices and the server, respectively. During forward propagation, intermediate feature maps (also called *smashed data*) from the cutting position (also called *cutlayer*) are transmitted from end devices to the server to facilitate training on the server-side layers. In backpropagation, partial derivative data from the cutlayer is distributed to the end devices to update client-side models. Consequently, the SL paradigm enables resource-limited end devices to participate in large-scale DNN training while protecting the privacy of both local data and the global model.

One of the primary challenges in SL is the high communication overhead [45] that results from transmitting feature maps, particularly during forward propagation. In FL, lazy aggregation [28] can linearly reduce communication overhead. However, due to the non-additive nature of feature maps, lazy aggregation cannot be directly applied to SL. As a result, iterative transmissions in the SL paradigm consume significant time and are unaffordable due to limited bandwidth connections between the end devices and the server. Therefore, it is essential to develop communication-efficient techniques for SL to enable its practical deployment in the edge environment.

Various technologies have been investigated to reduce communication complexity in FL and other distributed learning paradigms, typically including quantization [24, 40] and sparsification [40, 47]. However, these approaches are unsuitable for SL because they are designed to compress gradients or model parameters. Firstly, compressing feature maps, even with unbiased compression, will inevitably introduce biased gradients, thus affecting the model's convergence [2, 5, 36]. Secondly, compression error is a crucial determinant of the model's convergence. However, quantization and sparsification methods typically result in massive compression errors because quantization is sensitive to outliers (top values in the feature map), and sparsification produces errors due to filtered values. Finally, since feature maps from different training samples are independent, existing compensation mechanisms [8, 30], such as reserving and adding filter-out values after Top-k sparsification for the subsequent transmission, are inapplicable.

To overcome the abovementioned challenges, we propose a new communication compression framework for SL: mask-encoded sparsification (MS). MS is based on Top-k sparsification and uses an encoded mask to compensate for the significant compression errors caused by sparsification. Specifically, the feature map is first subject to a Top-k sparsification, after which an encoded mask indicates the position and value of every element in the feature map. The mask of all top values is a binary number of all 1s, and the remaining masks are calculated with a quantization-like method. As depicted in Fig. 1, MS utilizes a narrow bit width mask (2-bit), effectively reducing the compression error caused by sparsification and being helpful for convergence.

The main contributions of this work are summarized as follows:

- Our study reveals that compressing feature maps leads to biased gradients in SL. We establish the convergence rate of SL with compressed feature maps and find a situation where gradients converge to 0 at the same rate as in the uncompressed version.
- Our theoretical findings indicate that lower compression errors lead to improved convergence. Therefore, we propose mask-encoded sparsification (MS) and theoretically demonstrate its superior performance over previous methods under mild conditions. Also, empirical validation confirms that MS achieves lower errors than previous approaches.

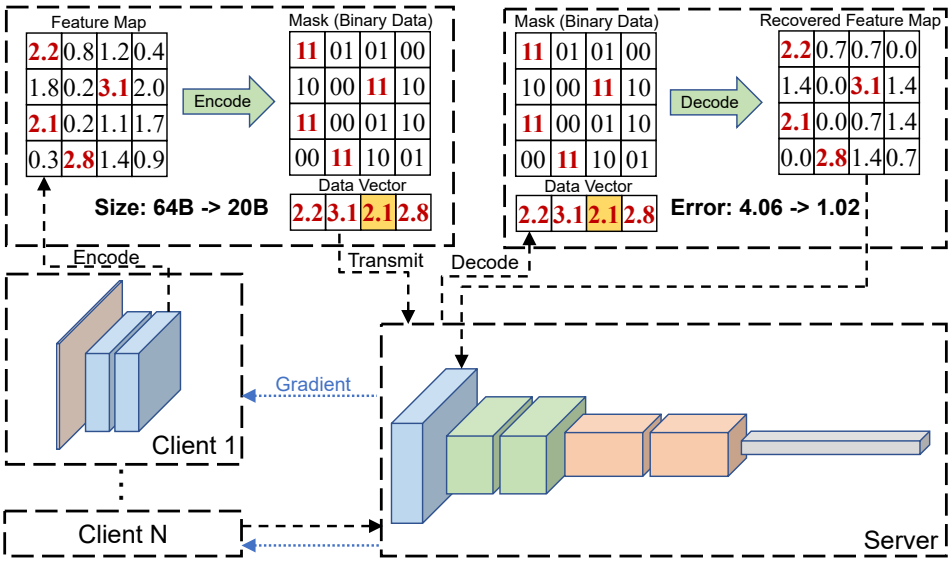

Figure 1: Using MS algorithm to compress smashed data in SL. Perform Top-4 sparsification on the smashed data, then compensate for filtered data with a 2-bit width mask. The top values are highlighted in red and will form a data vector in their original order. Their corresponding mask is $0b11$. The filtered values are calculated with a quantization-like method, MS evenly divides 0 to 2.1 (the smallest top value) into 3 ($3 = 2^2 - 1$) segments ($[0, 0.7)$,$[0.7, 1.4)$, and $[1.4, 2.1)$), the value in each segment will be rounded to 0, 0.7, and 1.4, and their masks are $0b00$, $0b01$, and $0b10$, respectively. Finally, the data vector is sent to the server along with the mask for decoding. All values with a mask of $0b11$ will be sequentially filled with the Top value, while the remaining masks will be filled with 0, 0.7, and 1.4, respectively. This strategy markedly diminishes compression errors (measured by the ℓ_2 norm) from 4.06 to 1.02, achieving superior performance compared to vanilla sparsification.

- We conduct extensive experiments on various models and datasets. Experimental results show that MS significantly reduces communication overhead without accuracy loss, while previous methods deteriorate convergence or generalization. Notably, according to the empirical study on different cutlayers, we find that the feature extraction layers of neural networks (i.e., shallow layers) are more sensitive to compression errors.

2 Related Work

2.1 Split Learning

Split Learning (SL) [7, 15, 16, 35] partitions a DNN into client-side and server-side model [37, 38, 43, 14], which are deployed on multiple end devices and the server, respectively. This strategy addresses not only the efficient utilization of distributed computing power but also privacy concerns related to raw data and the sharing of the global model [38], as it allows clients to pre-process their data locally and retain a part of the model. Despite these benefits, SL still encounters the challenge of communication overhead [44, 45]. The frequent exchange of local feature maps between the client and server leads to significant network traffic, especially in wireless networks, which can negatively impact training efficiency.

2.2 Communication Compression

Biased Compression Techniques Quantization [10, 26] aims to minimize the bit width of numerical values by transforming them from their "continuous" forms to discrete counterparts, thus reducing the data storage demands. However, quantization inherently exhibits sensitivity toward outliers, leading to potentially significant compression errors. Methods based on clipping [20, 32] provide some relief by clipping these outliers, yet the core issue is that outliers in feature maps are essential and cannot merely be overlooked. Alternatives such as clustering quantization [1] and learning-based quantization strategies [12] show promise in diminishing these compression errors. However, such approaches' computational overhead and complexity render them impractical for high-frequency iterative training scenarios, where rapid and efficient computation is paramount. In addition, sparsification converts dense vectors into their sparse forms. A prevalent technique is the Top-k sparsification [4], which retains only a vector's most significant k values.

Unbiased Compression Techniques Unbiased random quantization [3] distinguishes itself through random rounding procedures, marking an advancement in quantization that guarantees the operation's unbiased nature. Similarly, there is also an unbiased variant of sparsification [41] by enlarging randomly retained values, which guarantees an unbiased sparsification process.

In the learning paradigm of transmission gradients, such as in FL, applying unbiased compression techniques has been shown to enable DNNs to converge and realize a convergence rate the same as uncompressed gradient [3, 41]. However, applying unbiased compression directly to feature maps in SL does not safeguard against the production of unbiased gradients. Additionally, these compression technologies tend to induce substantial compression errors, challenging their applicability for feature map compression in SL.

3 Feature Map Compression

This section presents the problem formulation associated with the feature map compression within the SplitFed learning paradigm [37]. We further provide an in-depth analysis of gradients and convergence when applying feature map compression techniques in SL.

3.1 Problem Formulation

Consider a SL framework consisting of N clients. Client i has a local model $f_i^c(\cdot)$ parameterized by θ_i^c, while its complementary server-

side model is denoted as $f_i^s(\cdot)$, parameterized by θ_i^s. Furthermore, client i has an unbiased sample dataset x_i and corresponding labels y_i.

During the forward propagation phase, client i transmits the smashed data, $f_i^c(\theta_i^c; x_i)$, along with the labels y_i to the server for further computations. The challenge lies in the transmission of $f_i^c(\theta_i^c; x_i)$, which is the primary bottleneck in the SL training process, primarily due to the generally slower uplink speeds than downlink speeds. We focus on compressing the smashed data, with the compression error at client i denoted by ϵ_i. The server computes the aggregate loss as follows:

$$L(\theta_1^c, \theta_1^s, \ldots, \theta_N^c, \theta_N^s; x_1, y_1, \epsilon_1, \ldots, x_N, y_N, \epsilon_N) \coloneqq \frac{1}{N}\sum_{i=1}^{N} f_i^s(\theta_i^s; f_i^c(\theta_i^c; x_i) + \epsilon_i, y_i). \quad (1)$$

During backpropagation, the server computes the gradient $\hat{g_i^s} = \nabla_{\theta_i^s} L$, followed by the mean gradient $\hat{g^s} = \frac{1}{N}\sum_{i=1}^{N} \hat{g_i^s}$ for model parameter updates. The gradient concerning each client's smashed data, $\nabla_{f_i^c(\theta_i^c; x_i+\epsilon_i)} L$, returns to the respective client i for parameter adjustment. Subsequently, client i calculates its parameter gradient, $\hat{g_i^c} = \nabla_{\theta_i^c} f_i^c(\theta_i^c; x_i + \epsilon_i) \nabla_{f_i^c(\theta_i^c; x_i+\epsilon_i)} L$. All clients then collaborate to aggregate these gradients through either a server-based or an all-reduce method, yielding $\hat{g^c} = \frac{1}{N}\sum_{i=1}^{N} \hat{g_i^c}$. Both clients and the server update their respective parameters employing stochastic gradient descent with a learning rate η:

$$\theta_i^c \coloneqq \theta_i^c - \eta \cdot \hat{g^c}, \quad \theta_i^s \coloneqq \theta_i^s - \eta \cdot \hat{g^s}. \quad (2)$$

In scenarios where no compression is applied to the feature maps ($\epsilon_i = 0$), we denote the server gradient, client gradient, aggregated server gradient, and aggregated client gradient as g_i^s, g_i^c, g^s, and g^c, respectively.

Given the aggregation of parameters in each round, the objective function can be simplified to:

$$F(\theta^c, \theta^s) \coloneqq L(\theta^c, \theta^s, \ldots, \theta^c, \theta^s), \quad (3)$$

thereby rendering the SL's optimization objective as:[1]

$$[\theta^c, \theta^s]^* \coloneqq \arg\min_{\theta^c, \theta^s} F(\theta^c, \theta^s). \quad (4)$$

3.2 Gradient Analysis

In FL, communication compression techniques such as unbiased random quantization and rand-k sparsification are commonly utilized to preserve the unbiased gradients at the parameter server. However, even when these unbiased compression methods are applied to feature maps within the SL framework, the gradients are biased.

We state this formally as follows:

Theorem 1 (Compressed Feature Maps Bring Biased Gradients)**.** *Even unbiased compression techniques ($\mathbb{E}(\epsilon_i) = 0$) are applied to feature maps in SL, the outcome inevitably leads to biased gradients:*

$$\mathbb{E}([\hat{g^c}, \hat{g^s}]) \neq [g^c, g^s] \quad (5)$$

Proof. Assume that the unbiased server-side gradient is achieved, i.e., $\mathbb{E}(\hat{g^s}) = g^s$. Denote the original feature map of client i as z_i and its compressed version as $\hat{z_i}$, with the expectation $\mathbb{E}(\hat{z_i}) = z_i$. The server-side gradient is expressed as a function of the feature map, denoted as $h(\cdot)$, such that for the original feature map we have $g_i^s = h(z_i)$, and for the compressed feature map, $\hat{g_i^s} = h(\hat{z_i})$. It follows that $\mathbb{E}(\hat{g_i^s}) = \mathbb{E}(h(\hat{z_i}))$ and $h(\mathbb{E}(\hat{z_i})) = h(z_i) = g_i^s$.

If $\mathbb{E}(\hat{g_i^s}) = g_i^s$, for any server-side gradient function $h(\cdot)$ and any feature map z:

$$\mathbb{E}(h(z)) = h(\mathbb{E}(z)). \quad (6)$$

Consider the widely used activation function ReLU, defined as $a(x) = \max(0, x)$, and let z be a random variable that adheres to a standard normal distribution. We find that $\mathbb{E}(a(z)) = \int_0^\infty x \frac{1}{\sqrt{2\pi}} \exp^{-\frac{x^2}{2}} dx > 0$ but $a(\mathbb{E}(z)) = a(0) = 0$. Therefore $\mathbb{E}(a(z)) \neq a(\mathbb{E}(z))$. This observation contradicts Eq. (6) because DNN typically includes many activation functions.

Thus, the assumption does not hold, which means that we can not obtain an unbiased estimate of the server-side gradient $\mathbb{E}(\hat{g^s})$ under the compression: $\mathbb{E}(\hat{g^s}) \neq g^s$. Consequently, the expectation of the composite gradient $\mathbb{E}([\hat{g^c}, \hat{g^s}])$ does not equal the true gradients $[g^c, g^s]$. □

3.3 Convergence Analysis

In this section, we conduct a comprehensive convergence analysis of compressed feature maps in SL. Our analysis is based on assumptions commonly employed in distributed optimization [6, 13, 19, 27]:

Assumption 1 (Lipschitz Smoothness)**.** *There exists a finite positive constant L such that for all θ_1^c, θ_2^c in the client-side model and θ_1^s, θ_2^s in the server-side model, the Lipschitz condition is satisfied as follows:*

$$\|\nabla F(\theta_1^c, \theta_1^s) - \nabla F(\theta_2^c, \theta_2^s)\| \leq L\|[\theta_1^c, \theta_1^s] - [\theta_2^c, \theta_2^s]\|. \quad (7)$$

Assumption 2 (Unbiased Gradients)**.** *For every mini-batch, the stochastic gradients of the client-side and server-side models are unbiased estimators of the true gradient:*

$$\mathbb{E}([g^c, g^s]) = \nabla F(\theta^c, \theta^s). \quad (8)$$

Assumption 3 (Bounded Variance)**.** *The variances of the stochastic gradients in the client-side and server-side models are bounded by a finite positive constant σ^2:*

$$\mathbb{E}\|\nabla F(\theta^c, \theta^s) - [g^c, g^s]\|^2 \leq \sigma^2. \quad (9)$$

Given Assumptions 1-3, we establish the following lemma to bound the gradient errors in the client-side and server-side models.

Lemma 2 (Bounded Gradient Error)**.** *Let J and H denote the upper bounds of the Frobenius norms of the Jacobian matrix for the client-side model and the Hessian matrix for the server-side model, respectively. Define $E \coloneqq \frac{1}{N}\sum_{i=1}^{N} \|\epsilon_i\|$, where ϵ_i represents the compression error for the i-th client. The gap in the server-side gradient g^s and its compressed counterpart $\hat{g^s}$, as well as those in the client-side gradient g^c and $\hat{g^c}$, are bounded as follows:*

$$\|\hat{g^s} - g^s\| \leq HE, \quad \|\hat{g^c} - g^c\| \leq HJE. \quad (10)$$

Proof. Denote server-side hessian function as $H^s(\cdot)$. According to the Lagrange mean value theorem, there are a set of points $\xi_i (i =$

[1] In this paper, Use $[v_1, v_2]$ to represent column vectors v_1 and v_2 concatenated by columns.

$1, 2 \dots N$) yields:

$$\|\hat{g^s} - g^s\| \tag{11}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N}(\hat{g_i^s} - g_i^s)\| \tag{12}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N}(\nabla_{\theta_i^s} f^s(\theta_i^s; \hat{f_i^c}(\theta_i^c; x_i), y_i) - \nabla_{\theta_i^s} f^s(\theta_i^s; f_i^c(\theta_i^c; x_i), y_i))\| \tag{13}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} H^s(\xi_i)^\top(\hat{f_i^c}(\theta_i^c; x_i) - f_i^c(\theta_i^c; x_i))\| \tag{14}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} H^s(\xi_i)^\top \epsilon_i\| \tag{15}$$

$$\leq \frac{1}{N}\sum_{i=1}^{N} \|H^s(\xi_i)\|_F \|\epsilon_i\| \tag{16}$$

$$\leq HE. \tag{17}$$

Use identity matrix for approximate differentiation of compression operation, and follow the chain derivation rule:

$$\|\hat{g^c} - g^c\| \tag{18}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} \hat{g_i^c} - g^c\| \tag{19}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_i^c} f^s(\theta^s; f_i^c(\theta_i^c; x_i) + \epsilon_i) - g^c\| \tag{20}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} \nabla_{f_i^c(\theta_i^c; x_i)} f^s(\theta^s; f_i^c(x_i; \theta_i^c) + \epsilon_i)^\top \nabla_{\theta_i^c} f_i^c(\theta_i^c; x_i) - g^c\| \tag{21}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} (\nabla_{f_i^c(\theta_i^c; x_i)} f^s(\theta^s; f_i^c(\theta_i^c; x_i)) - g^c\| \tag{22}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_i^c} f^s(\theta^s; f_i^c(\theta_i^c; x_i)) + \frac{1}{N}\sum_{i=1}^{N} H^s(\xi_i)\epsilon_i^\top \nabla_{\theta_i^c} f_i^c(x_i; \theta_i^c) - g^c\| \tag{23, 24}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} g_i^c + \frac{1}{N}\sum_{i=1}^{N} H^s(\xi_i)\epsilon_i^\top \nabla_{\theta_i^c} f_i^c(x_i; \theta_i^c) - g^c\| \tag{25}$$

$$= \|\frac{1}{N}\sum_{i=1}^{N} H^s(\xi_i)\epsilon_i^\top \nabla_{\theta_i^c} f_i^c(x_i; \theta_i^c)\| \tag{26}$$

$$\leq \frac{1}{N}\sum_{i=1}^{N} \|H^s(\xi_i)\|_F \|\epsilon_i\| \|\nabla_{\theta_i^c} f_i^c(x_i; \theta_i^c)\| \tag{27}$$

$$\leq HJE. \tag{28}$$

Therefore, Lemma 2 is proven.

□

Building on Assumptions 1-3 and Lemma 2, we derive the convergence rate in SL as follows:

Theorem 3 (Convergence in SL)**.** *Under Assumptions 1-3 and Lemma 2, use subscripts to represent the number of iterations. Consider* $\varepsilon^2 = (1 + J^2)H^2 \frac{1}{T}\sum_{t=1}^{T} E_t$ *and* $\gamma = \mathbb{E}(F(\theta_1^c, \theta_1^s) - F(\theta_{T+1}^c, \theta_{T+1}^s))$*. By choosing the learning rate* $\eta = \sqrt{\frac{\gamma}{TL(\sigma^2+\varepsilon^2)}}$ *and ensuring that the inner product between the error gradient (e.g.,* $[\hat{g^c}, \hat{g^s}] - [g^c, g^s]$*) and the full gradient (e.g.,* $\nabla_{[\theta^c, \theta^s]} F(\theta^c, \theta^s)$*) is always non-negative, with* $\eta \in (0, \frac{1}{2L}]$*, SL can achieve the following convergence rate:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla F(\theta_t^c, \theta_t^s)\|^2 \leq 4\sqrt{\frac{\gamma L(\sigma^2 + \varepsilon^2)}{T}}. \tag{29}$$

In other scenarios, with $\eta \in (0, \frac{1}{4L}]$*, the convergence rate is given by:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla F(\theta_t^c, \theta_t^s)\|^2 \leq 8\sqrt{\frac{\gamma L(\sigma^2 + \varepsilon^2)}{T}} + 2\varepsilon^2. \tag{30}$$

Proof. Please refer to the Appendix A for the proof of Theorem 3. □

Remark 1. *Theorem 1 indicates the effect of feature map compression on gradient bias within the SL framework. This phenomenon persists even when employing ostensibly unbiased compression methodologies, and such a biased gradient hurts the model's convergence. Complementing this, Theorem 3 delineates how the compression errors directly influence the convergence rates of both the client and server-side models. These conclusions indicate that previous compression techniques, such as quantization and sparsification, are unsuitable for SL because they can not obtain unbiased gradients and introduce substantial errors.*

4 Mask-Encoded Sparsification

In this section, we introduce the mask-encoded sparsification (MS) algorithm and provide theoretical and experimental evidence to demonstrate that the algorithm's error is lower than that of traditional methods.

4.1 Algorithm Details

Inspired by the max pooling algorithm, we believe that the top values within a feature map are critical in DNNs. However, sparsification leads to significant compression errors, especially with a higher sparsification ratio. As pointed out in Theorem 3, this can negatively affect the model's convergence. To address this, we propose a new method to weaken the values lost due to sparsification, thus reducing the impact of compression errors.

For vanilla Top-k sparsification, we use a mask that matches the size of the feature map. We set bits in the mask to $0b1$ for the top values and $0b0$ for the filtered values, storing all top values in order. During the decoding process, we traverse the mask and sequentially fill in the top values wherever the mask has a bit $0b1$, replacing all other values with $0b0$. This sparse matrix storage method costs less than the key-value pair storage method when the sparsification ratio is below 96.875% (Please refer to the Appendix B for more details).

In MS, we expand the mask bit width to b and assign a binary number composed of b ones to the position mask of the top values. This approach allows the additional bits to compensate for the values filtered by Top-k sparsification. Additionally, we assume the feature map contains only positive values (applying the algorithm after the ReLU function). If the feature map includes negative numbers, we add a bit to the mask to record symbol information. Consequently, all filtered values range from 0 to the smallest top value, allowing

us to divide this range into $2^b - 1$ segments evenly. For a filtered value t, the corresponding mask position records the binary form of $\lfloor t \times (2^b - 1)/Top_{\min} \rfloor$, where $Top_{\min}$ is the smallest top value. The client transmits top values and the encoded mask to the server.

Upon receiving this data, the server first identifies the minimum top value $Top_{\min}$. Then, it iterates through the mask. Whenever it encounters a number where all bits are 1, the server pops the first value from the data vector and assigns it to the corresponding position. For other mask values, the server multiplies the mask value by $Top_{\min}/(2^b - 1)$ to reconstruct the filtered value. For a detailed explanation of this process, refer to Algorithm 1 and Algorithm 2.

Algorithm 1 Encoding Stage of Mask-Encoded Sparsification

Input: d-dimensional feature map x
Parameter: Sparsification ratio r, mask bit width b
Output: Vector of top values v, encoded mask m

Conduct Top-k sparsification on x, with $k = \lfloor (1 - r) \times d \rfloor$. Record the smallest top value as $Top_{\min}$ and arrange the top values in order to form vector v.
Initialize a d-dimensional mask with b bits per position.
for $i = 1$ to d **do**
  if $x_i \geq Top_{\min}$ **then**
    $m_i \leftarrow 2^b - 1$.
  else
    $m_i \leftarrow \lfloor x_i \times (2^b - 1)/Top_{\min} \rfloor$.
  end if
end for
return vector v and mask m.

Algorithm 2 Decoding Stage of Mask-Encoded Sparsification

Input: Vector of top values v, encoded mask m
Parameter: Mask bit width b, dimension of feature map d
Output: Decoded feature map $\hat{x}$

Initialize $\hat{x}$ with the same dimension as m.
Retrieve the minimum top value $Top_{\min}$ from v.
for $i = 1$ to d **do**
  if $m_i = 2^b - 1$ **then**
    Pop the first value t from v.
    $\hat{x}_i \leftarrow t$.
  else
    $\hat{x}_i \leftarrow m_i \times Top_{\min}/(2^b - 1)$.
  end if
end for
return the reconstructed feature map $\hat{x}$.

In Fig. 1, the client generates a 2-bit mask with the same dimensions as the feature map. Subsequently, Top-4 sparsification is applied to the feature map, resulting in a data vector containing the top values in their original order. The corresponding mask position is assigned a binary number with all 1 bits (here is $0b11$). The smallest top value recorded during this process is 2.1. Additionally, the client uniformly maps all filtered values within the $[0, 2.1]$ range to integers ranging from 0 to 3 (where $3 = 2^2 - 1$). These mapped values are converted into binary numbers and placed within the corresponding mask. Upon receiving the encoded mask and data vector, the server performs the following steps to decode the data. Firstly, it identifies the smallest top value of 2.1. Next, it iterates through the mask. If a mask value of all $0b11$ is encountered, the server pops the first element from the data vector and fills it into the corresponding position. Otherwise, the mask value is mapped to the interval $[0, 2.1]$ and filled into the recovered feature map. By following this process, the server successfully obtains the decoded data.

MS effectively compresses the data size through this approach, reducing communication overhead. The example illustrated in Fig. 1 demonstrates a reduction in traffic from 64 bytes to 20 bytes compared to uncompressed data. Furthermore, it decreases the 2-norm compression error from 4.06 to 1.02 compared to vanilla Top-k sparsification.

4.2 Error Analysis

We compare the error rates of our new algorithm (MS) with a couple of existing ones: Top-k sparsification (SP) [4, 42, 47], quantization (QU) [3, 10, 26], and randomized Top-k sparsification (RT) [44].

Error upper bound comparison In the RT method, sparsification is achieved by repeated sampling, with the top values having a higher probability of being selected. It can be proven that RT introduces more compression errors than SP. Therefore, we do not consider the compression error between RT and MS.

Assuming that the feature map is x of length d, where each value has a bit width of f, the values in QU are mapped to a bit width of q_1. In SP, k_1 top values are retained, while in the MS approach, k_2 top values are kept, utilizing a q_2-bit mask. The compression ratios of these three algorithms, QU, SP, and MS, are equal when the conditions $q_1 d = q_2 d + f k_2$ and $d + f k_1 = q_2 d + f k_2$ are met.

The error upper bounds for QU and SP are defined by Eq. (31) [3] and Eq. (32) [4], respectively. In our method, the error primarily arises from the compensation of values filtered out by SP. Consequently, the upper bound of the compression error in MS adheres to Eq. (33). Here, α represents the ratio of the 2-norm of the values filtered by SP to the 2-norm of x.

$$\mathbb{E}\|QU(x) - x\|^2 \leq \sqrt{d}/(2^{q_1} - 1)\|x\|^2, \tag{31}$$

$$\mathbb{E}\|SP(x) - x\|^2 \leq (d - k_1)/d\|x\|^2, \tag{32}$$

$$\mathbb{E}\|MS(x) - x\|^2 \leq \alpha\sqrt{d - k_2}/(2^{q_2} - 1)\|x\|^2. \tag{33}$$

Theorem 4 (The condition for better than QU and SP)**.** *With the increase of q_2, the upper bound error of MS is less than SP. When $\alpha \in (0, 1/2)$ and $k_2/d \to 0$, the upper bound error of MS is less than QU.*

Proof. Please refer to the Appendix C for the proofs of Theorem 4. □

Error comparison experiment We examined the compression errors of feature maps on CIFAR-10, CIFAR-100, and ImageNet-1K, employing SP, QU, MS, and RT algorithms at identical compression levels. The comparison is illustrated in Fig. 2.

Notably, the compression error of MS is always lower than QU, SP, and RT. MS compensates for the values filtered out by SP through the mask, resulting in an error lower than SP. Additionally, MS preserves top values, thus avoiding the sensitivity of quantization to outliers, and the error is also lower than quantization. The experiment also proves that the compression error of RT is consistently higher than that of SP at the same compression rate.

In addition, the experiment also verified the Theorem 4 that as q_2 increases (compression rate decreases), the gap between the compression error of MS and SP gradually increases. When the sparsification rate increases (compression rate increases), the compression error of MS is lower than that of QU.

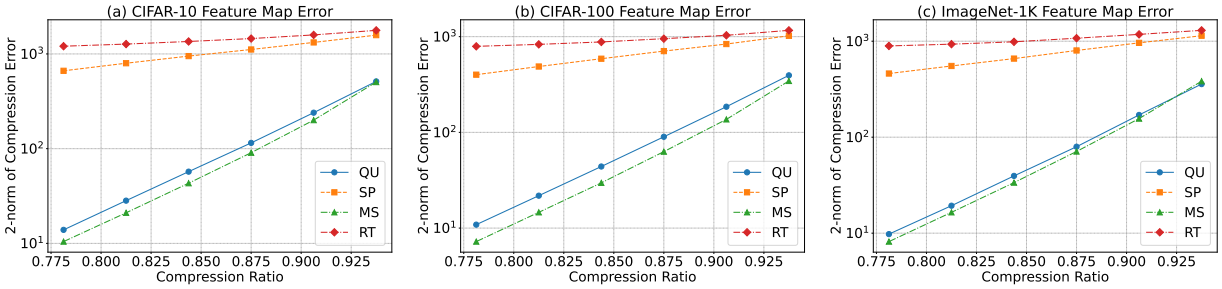

Figure 2: The average 2-norm compression error of feature maps for QU, SP, MS, and RT on different datasets as the compression rate increases. Figures (a), (b), and (c) show the comparison of CIFAR-10, CIFAR-100, and ImageNet-1K, respectively.

Remark 2. *Building on Theorems 4, it is observed that MS achieves a lower upper bound on the compression error compared to QU and SP. This performance improvement is noted under conditions of increased sparsification ratio and expanded mask bit width, which are relatively mild constraints. Furthermore, Fig. 2 illustrates that MS consistently outperforms QU, SP and RT regarding compression error. The theoretical insights and empirical results consistently underscore the superiority of MS over QU, SP, and RT in terms of compression error. This advantage facilitates improved convergence in SL.*

4.3 Time Complexity

During the encoding phase, SP exhibits a time complexity of $\mathcal{O}(d\log(k))$, attributed to the Top-k operation. Here, d denotes the feature map length, and k represents the retained top values. Both MS and RT are improved versions based on SP. MS introduces an additional operation of mask encoding, with a time complexity of $\mathcal{O}(d)$, so the total time complexity of MS is $\mathcal{O}(d\log(k)+d)$. However, RT introduces k sampling operations, where the time complexity of each sampling is $\mathcal{O}(d)$, and the total time complexity of RT is $\mathcal{O}(d\log(k)+dk)$. Consequently, MS provides powerful performance without increasing the complexity of the time order. In contrast, RT increased the time complexity of SP from $\mathcal{O}(d\log(k))$ to $\mathcal{O}(dk)$, which affected the training efficiency of SL.

Some compression methods, such as clustering quantization [1] and learning-based quantization [12], may achieve lower compression errors than MS but at the expense of a higher time complexity during both encoding and decoding phases. This drawback renders them less suitable for SL, where iterative training is frequent and time efficiency is paramount.

5 Experiments

5.1 Experimental Setup

Devices To simulate a realistic edge environment, we use HUAWEI Atlas DK200 as the client, connected to an NVIDIA GeForce RTX 3090 server via a 100Mbps wireless network.

Benchmarks We train three models on three datasets: VGG19 [33] on CIFAR-10 [22], ResNet18 [17] on CIFAR-100 [22], and ResNet34 [17] on ImageNet-1K [31]. We evenly cut the three datasets into ten parts and put them into ten clients. For these three models, we apply three cutting strategies, which are shallow, medium, and deep cuts. Specifically, for VGG19, we cut the model after layers 2, 8, and 15, respectively; for ResNet18, we cut the model after layers 2, 9, and 13, respectively; and for ResNet34, we cut the model after layers 2, 15, and 27, respectively.

Compression Policies When using MS compression, the sparsification ratio is 99%, the bit width is 2, the compression rate is 92.75%. We compare 95.875% SP and 95.875% RT at our MS's compression level. Due to the unsmooth compression ratio of quantization, we use 3-bit quantization with a compression rate of 90.625%. We also recorded the uncompressed version as the baseline.

Metrics We compare the convergence and generalization of the models by analyzing the average training loss and test accuracy curves as the epoch increases. In addition, we also compare the traffic multiples saved by all compression strategies to achieve baseline accuracy.

5.2 Experimental Results

Convergence and Generalization The convergence and generalization performance for VGG19, ResNet18, and ResNet34 are illustrated in Fig. 3, respectively. In these figures, the solid black line indicates the baseline trend, the blue dotted line depicts the changes in QU, the green stippled line shows the changes in SP, and the red dotted line represents the changes in MS. Additionally, square, diamond, and triangular markers correspondingly denote shallow, middle, and deep cutlayers.

As depicted in Fig. 3, it is observed that in most situations, QU, SP, and struggle to converge to the baseline performance level and fail to achieve baseline accuracy (only medium and deep cuts in VGG19, QU, and SP can reach the baseline level). RT can converge better than MS in certain situations (ResNet18 and ResNet34), but its generalization is still poor. These phenomenons underscore the limitations of QU, SP, and RT in the context of feature map compression and are due to the vast compression errors induced by QU, SP, and RT. On the contrary, MS can match the baseline in terms of convergence and generalization in most situations (only the situation of shallow cut in ResNet34 is slightly lower than the baseline level, and its accuracy is approximately 0.5% less) because MS ensures the integrity of the top value and achieves lower compression error. These results highlight the superiority of the MS algorithm in compressing feature maps.

We also discover an interesting phenomenon: all compression algorithms' convergence and generalization abilities tend to improve as the cutting layer deepens. This phenomenon suggests that the shallow layers of neural networks are more sensitive to errors, whereas the deeper layers exhibit greater error tolerance.

Communication Efficiency We record the communication traffic needed to reach the baseline test accuracy in all experiments, as shown in Table 1. Bold values for communication traffic indicate the minor overhead each compression policy requires to meet the target test accuracy. We mark cases where an experiment does not reach the target accuracy under the communication compression policy with "INF".

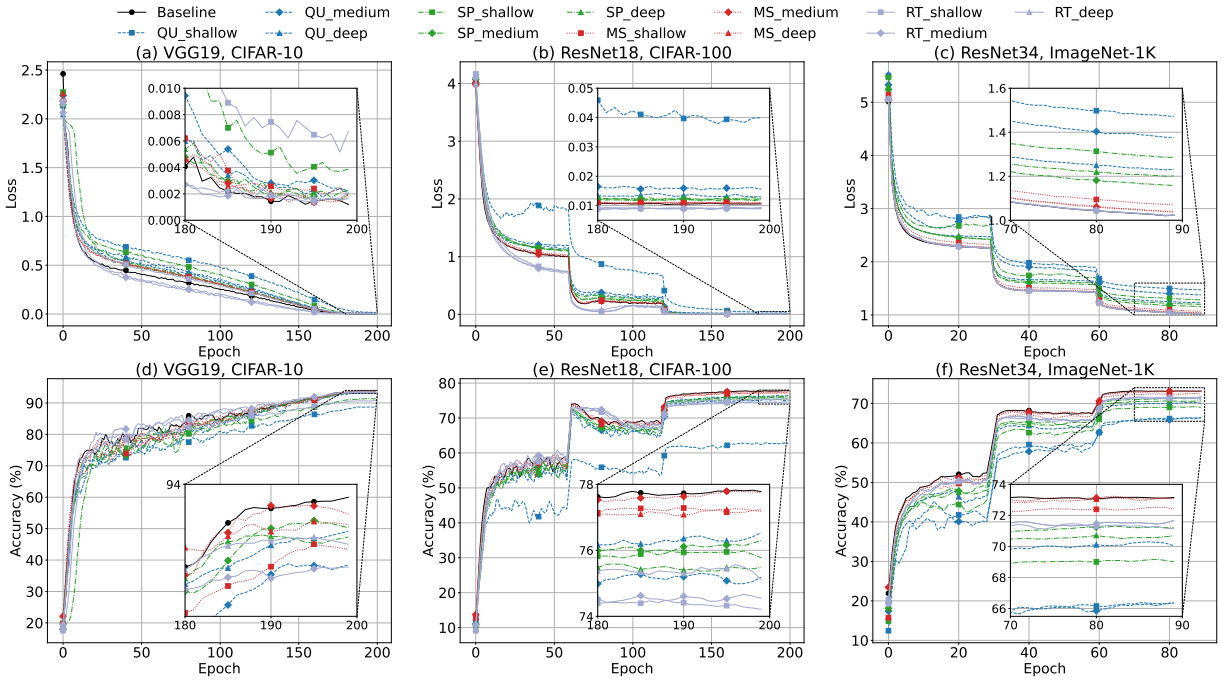

Figure 3: The experiment results of convergence and generalization. The first and second rows represent the changes in training loss and testing accuracy, respectively. The first, second, and third columns represent the results of CIFAR-10, CIFAR-100, and ImageNet-1K, respectively. The solid black line indicates the baseline trend, the blue dotted line depicts the changes in QU, the green stippled line shows the changes in SP, and the red dotted line represents the changes in MS. Additionally, square, diamond, and triangular markers correspondingly denote shallow, middle, and deep cutlayers.

Table 1: Comparison of communication traffic savings achieved by various compression algorithms compared to non-compression baseline, with "INF" indicating failure to achieve baseline accuracy level.

Datasets	Cutlayers	Traffic Saving Multiplier			
		QU	SP	MS	RT
CIFAR-10	Shallow	INF	INF	**13.11**×	INF
	Medium	9.92×	**13.56**×	**13.56**×	13.71×
	Deep	10.37×	**13.56**×	13.63×	13.71×
CIFAR-100	Shallow	INF	INF	**12.53**×	INF
	Medium	INF	INF	**13.50**×	INF
	Deep	INF	INF	**12.29**×	INF
ImageNet-1K	Shallow	INF	INF	INF	INF
	Medium	INF	INF	**13.20**×	INF
	Deep	INF	INF	**12.32**×	INF

Overall, QU, SP and RT often fail to achieve baseline accuracy and can only converge on less complex datasets such as CIFAR-10. This is because they bring higher compression errors. MS consistently meets the baseline accuracy in nearly all tests. Compared to the other policies, MS typically requires minor communication traffic, except in the shallow cutlayer of ResNet34, where it achieved about 72.6% accuracy (0.5% lower than baseline). These results highlight MS's efficiency in ensuring convergence and significantly reducing communication overhead.

6 Conclusion

In this work, we show that compressed feature maps introduce biased gradients in SL and provide a comprehensive convergence analysis in the face of feature map compression. Our findings confirm the critical role of compression error in the convergence process, revealing the inherent limitations of existing compression methodologies within SL frameworks. To address these shortcomings, we propose the MS algorithm, which employs a narrow bit width mask to mitigate sparsification errors, a strategy that keeps accuracy without adding algorithmic complexity. Our theoretical and empirical evidence shows that the MS algorithm outperforms conventional techniques, effectively diminishing compression errors while maintaining the same compression level. Extensive experiments on various DNN models and datasets also demonstrate the effectiveness and efficiency of MS.

References

[1] A. Abernathy and M. E. Celebi. The incremental online k-means clustering algorithm and its application to color quantization. *Expert Systems with Applications*, 207, 2022.

[2] A. Ajalloeian and S. U. Stich. On the convergence of SGD with biased gradients. *arXiv preprint arXiv:2008.00051*, 2020.

[3] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems (NeurIPS 2017)*, pages 1707–1718, 2017.

[4] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli. The convergence of sparsified gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS 2018)*, pages 5977–5987, 2018.

[5] A. Beznosikov, S. Horváth, P. Richtárik, and M. Safaryan. On biased compression for distributed learning. *Journal of Machine Learning Research*, 24(276):1–50, 2023.

[6] T. J. Castiglia, A. Das, S. Wang, and S. Patterson. Compressed-VFL: Communication-efficient learning with vertically partitioned data. In *International Conference on Machine Learning (ICML 2022)*, pages 2738–2766, 2022.

[7] Z. Cheng, X. Xia, M. Liwang, X. Fan, Y. Sun, X. Wang, and L. Huang. CHEESE: Distributed clustering-based hybrid federated split learning over edge networks. *IEEE Transactions on Parallel and Distributed Systems*, 34(12):3174–3191, 2023.

[8] L. Condat, K. Yi, and P. Richtarik. Ef-bv: A unified theory of error feedback and variance reduction mechanisms for biased and unbiased

compression in distributed optimization. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*, pages 17501–17514, 2022.

[9] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin, R. Jenatton, L. Beyer, M. Tschannen, A. Arnab, X. Wang, C. Riquelme Ruiz, M. Minderer, J. Puigcerver, U. Evci, M. Kumar, S. V. Steenkiste, G. F. Elsayed, A. Mahendran, F. Yu, A. Oliver, F. Huot, J. Bastings, M. Collier, A. A. Gritsenko, V. Birodkar, C. N. Vasconcelos, Y. Tay, T. Mensink, A. Kolesnikov, F. Pavetic, D. Tran, T. Kipf, M. Lucic, X. Zhai, D. Keysers, J. J. Harmsen, and N. Houlsby. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning (ICML 2023)*, pages 7480–7512, 2023.

[10] R. Dorfman, S. Vargaftik, Y. Ben-Itzhak, and K. Y. Levy. DoCoFL: Downlink compression for cross-device federated learning. In *International Conference on Machine Learning (ICML 2023)*, pages 8356–8388, 2023.

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR 2021)*, 2021.

[12] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *IEEE/CVF international conference on computer vision (ICCV 2019)*, pages 4852–4861, 2019.

[13] K. Gruntkowska, A. Tyurin, and P. Richtárik. Ef21-p and friends: Improved theoretical communication complexity for distributed optimization with bidirectional compression. In *International Conference on Machine Learning (ICML 2023)*, pages 11761–11807, 2023.

[14] O. Gupta and R. Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[15] D.-J. Han, D.-Y. Kim, M. Choi, C. G. Brinton, and J. Moon. SplitGP: Achieving both generalization and personalization in federated learning. In *IEEE Conference on Computer Communications (INFOCOM 2023)*, 2023.

[16] D.-J. Han, D.-Y. Kim, M. Choi, D. Nickel, J. Moon, M. Chiang, and C. G. Brinton. Federated split learning with joint personalization-generalization for inference- stage optimization in wireless edge networks. *IEEE Transactions on Mobile Computing*, pages 1–17, 2023.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR 2016)*, pages 770–778, 2016.

[18] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1): 615–629, 2017.

[19] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning (ICML 2020)*, pages 5132–5143, 2020.

[20] H.-B. Kim, E. Park, and S. Yoo. Basq: Branch-wise activation-clipping search quantization for sub-4-bit neural networks. In *European Conference on Computer Vision (ECCV 2022)*, pages 17–33. Springer, 2022.

[21] S. Krause and F. Stolzenburg. Commonsense reasoning and explainable artificial intelligence using large language models. In *European Conference on Artificial Intelligence (ECAI 2023))*, pages 302–319, 2023.

[22] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, 2009.

[23] T. Krüger and M. Gref. Performance of large language models in a computer science degree program. In *European Conference on Artificial Intelligence (ECAI 2023))*, pages 409–424, 2023.

[24] X. Li and P. Li. Analysis of error feedback in federated non-convex optimization with biased compression: Fast convergence and partial participation. In *International Conference on Machine Learning (ICML 2023)*, pages 19638–19688, 2023.

[25] X. Li, Z. Song, and J. Yang. Federated adversarial learning: A framework with convergence analysis. In *International Conference on Machine Learning (ICML 2023)*, pages 19932–19959, 2023.

[26] H. Liu, F. He, and G. Cao. Communication-efficient federated learning for heterogeneous edge devices based on adaptive gradient quantization. In *IEEE Conference on Computer Communications (INFOCOM 2023)*, pages 1–10, 2023.

[27] I. Markov, A. Vladu, Q. Guo, and D. Alistarh. Quantized distributed training of large models with convergence guarantees. In *International Conference on Machine Learning (ICML 2023)*, 2023.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics (AISTATS 2017)*, pages 1273–1282, 2017.

[29] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*, pages 27730–27744, 2022.

[30] X. Qian, P. Richtarik, and T. Zhang. Error compensated distributed SGD can be accelerated. In *Advances in Neural Information Processing Systems (NeurIPS 2021)*, pages 30401–30413, 2021.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[32] C. Sakr, S. Dai, R. Venkatesan, B. Zimmer, W. Dally, and B. Khailany. Optimal clipping and magnitude-aware differentiation for improved quantization-aware training. In *International Conference on Machine Learning (ICML 2022)*, pages 19123–19138, 2022.

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR 2015)*, 2015.

[34] Z. Song, Y. Wang, Z. Yu, and L. Zhang. Sketching for first order method: Efficient algorithm for low-bandwidth channel and vulnerability. In *International Conference on Machine Learning (ICML 2023)*, pages 32365–32417, 2023.

[35] V. Stephanie, I. Khalil, and M. Atiquzzaman. Digital twin enabled asynchronous splitfed learning in e-healthcare systems. *IEEE Journal on Selected Areas in Communications*, 41(11):3650–3661, 2023.

[36] A. A. S. U. Stich. Analysis of SGD with biased gradient estimators. *arXiv preprint arXiv:2008.00051*, 2020.

[37] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun. SplitFed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2022)*, pages 8485–8493, 2022.

[38] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[39] H. Wang, T. Fu, Y. Du, W. Gao, K. Huang, Z. Liu, P. Chandak, S. Liu, P. Van Katwyk, A. Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[40] J. Wang, Y. Lu, B. Yuan, B. Chen, P. Liang, C. De Sa, C. Re, and C. Zhang. CocktailSGD: Fine-tuning foundation models over 500Mbps networks. In *International Conference on Machine Learning (ICML 2023)*, pages 36058–36076, 2023.

[41] J. Wangni, J. Wang, J. Liu, and T. Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems (NeurIPS 2018)*, pages 1306–1316, 2018.

[42] H. Xu, W. Zhang, J. Fei, Y. Wu, T. Xie, J. Huang, Y. Xie, M. Elhoseiny, and P. Kalnis. SLAMB: Accelerated large batch training with sparse communication. In *International Conference on Machine Learning (ICML 2023)*, pages 38801–38825, 2023.

[43] B. Yin, Z. Chen, and M. Tao. Predictive gan-powered multi-objective optimization for hybrid federated split learning. *IEEE Transactions on Communications*, 71(8):4544–4560, 2023.

[44] F. Zheng, C. Chen, L. Lyu, and B. Yao. Reducing communication for split learning by randomized top-k sparsification. In *International Joint Conference on Artificial Intelligence (IJCAI 2023)*, pages 4665–4673, 2023.

[45] Q. Zhou, S. Guo, Y. Liu, J. Zhang, J. Zhang, T. Guo, Z. Xu, X. Liu, and Z. Qu. Hierarchical channel-spatial encoding for communication-efficient collaborative learning. In *Advances in Neural Information Processing Systems (NeurIPS 2022)*, pages 5788–5801, 2022.

[46] W. Zhou, Z. Qu, Y. Zhao, B. Tang, and B. Ye. An efficient split learning framework for recurrent neural network in mobile edge environment. In *Proceedings of the Conference on Research in Adaptive and Convergent Systems (RACS 2022)*, page 131–138, 2022.

[47] Y. Zhou, M. Shi, Y. Li, Y. Sun, Q. Ye, and J. Lv. Communication-efficient federated learning with single-step synthetic features compressor for faster convergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2023)*, pages 5031–5040, 2023.