Model Parallelism With Subnetwork Data Parallelism

Vaibhav Singh [*1 2] **Zafir Khalid** [*1 2] **Eugene Belilovsky** [1 2] **Edouard Oyallon** [3]

Abstract

Distributed pre-training of large models at scale often imposes heavy memory demands on individual nodes and incurs significant intra-node communication costs. We propose a novel alternative approach that reduces the memory requirements by training small, structured ***subnetworks*** of the model on separate workers. Unlike pipelining, our method avoids inter-node activation communication and maintains bandwidth requirements that are comparable to or lower than standard data parallel communication schemes based on all-reduce. We evaluate two subnetwork construction strategies guided by the principle of ensuring uniform representation of each parameter across the distributed training setup. Our results show that the stochastic block dropping technique consistently outperforms the width-wise subnetwork construction previously explored in federated learning. We empirically attribute this superior performance to stronger gradient alignment in subnetworks that retain blocks having skip connections. Preliminary experiments highlight the promise of our approach, achieving a $\mathbf{20}$–$\mathbf{40}\%$ reduction in memory usage without any loss in performance.

1. Introduction

The rapid scaling of deep neural networks has led to unprecedented progress across a wide range of domains, from computer vision (He et al., 2016a; Radford et al., 2021; Oquab et al., 2023; Kirillov et al., 2023; Shang et al., 2024) to natural language processing (Bommasani et al., 2021; Achiam et al., 2023; Touvron et al., 2023; Zhao et al., 2023a). Training such large models has necessitated distributed strategies like *data parallelism* (Li et al., 2020) and *model parallelism* (Shazeer et al., 2018; Shoeybi et al., 2019; Huang et al., 2019), each with trade-offs. Data parallelism, especially in the form of Distributed Data Parallel (DDP) (Li et al., 2020), replicates the model on each GPU and synchronizes gradients after each backward pass. While simple and widely adopted, it incurs high memory overhead due to full model replication and high communication cost during synchronization. Model parallelism (e.g., GPipe (Huang et al., 2019)) mitigates memory issues by splitting layers across devices but requires expensive high-bandwidth interconnects to communicate activations. Unlike data parallelism, where several techniques effectively reduce communication costs (Douillard et al., 2023a; Wang et al., 2023), reducing the bandwidth constraints when communicating activations is an open problem. Additionally, classic pipeline-based approaches often suffer from inefficiencies due to idle waiting (pipeline bubbles).

In this work, we explore *Subnetwork Data Parallelism* as a strategy to reduce per-node memory usage by fully distributing the training of model sub-components across devices. Unlike pipelining, which divides computation across sequential stages, our approach assigns each worker a ***subnetwork***, defined as a structurally complete portion of the model with a full path from input to loss, enabling independent gradient computation without exchanging activations. Each worker optimizes only its subset of parameters and synchronizes overlapping parameters through averaging at each step. Instead of replicating or fully sharding the model, we distribute these subnetwork s across nodes, so that each device hosts only a fraction of the full model. Each subnetwork is trained independently, with synchronization performed via parameter averaging to form a unified model. Since each node hosts only a fraction of the full model, overall memory usage is significantly reduced. Intra-node parallelism follows standard data parallelism with parameter-level communication, and the approach is compatible with existing systems-level model parallelism techniques.

Subnetwork training draws on two key insights: (i) large models often contain significant parameter redundancy, making it possible to estimate the loss with subnetworks; and (ii) partial training acts as a form of regularization, improving generalization. By training overlapping submodels and synchronizing only shared parameters, our framework balances scalability and efficiency. This yields a substantial reduction in memory footprint, as each GPU hosts only a subnetwork rather than the full model. By relaxing the need

[*]Equal contribution [1]Mila - Quebec AI Institute [2]Concordia University, Montréal [3]ISIR – Sorbonne Universite, Paris. Correspondence to: Zafir Khalid <zafirmk0@gmail.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

for full replication and reducing synchronization overhead, our approach improves both memory efficiency.

We formalize this approach as a distributed training framework in which each worker optimizes a fixed subset of model parameters, referred to as its ***subnetwork***. Each parameter is assigned to a fixed set of $P \leq N$ workers, ensuring that all parameters are updated by at least one worker. Following each local update, parameter replicas are averaged and broadcasted to maintain coherence across the global model. This strategy, which we term ***Subnetwork Data Parallelism***, avoids full model replication and pipeline dependencies, enabling efficient training on memory-constrained accelerators.

- We propose a distributed training paradigm—*Subnetwork Data Parallelism*, based on overlapping parameter assignment and averaging, enabling memory efficient training.
- We explore two subnetwork construction strategies: (i) selecting subsets of neurons or channels, and (ii) removing entire layers (blocks) from the network.
- We demonstrate that our approach achieves competitive performance on image classification tasks, while substantially reducing per-device memory usage and synchronization overhead.

2. Related Work

We now review the related work in model parallelism and in federated learning.

Pipeline parallelism Pipeline parallelism addresses memory bottlenecks by dividing the model itself across devices. In (Huang et al., 2019; Rivaud et al., 2024), the model is partitioned layer-wise across multiple devices, and micro-batches are pipelined to maximize hardware utilization. Mesh-TensorFlow (Shazeer et al., 2018) and Megatron-LM (Shoeybi et al., 2019) explore intra-layer tensor-splitting, where weights and activations are sharded across devices. While these methods allow overcoming memory constraints, they require multiple well-connected accelerators and also experience pipeline bubbles and load imbalance.

Another model parallelism direction has considered parallel learning of layers through auxiliary local losses (Belilovsky et al., 2020).

Fully Sharded and Zero Redundancy Approaches. To address memory inefficiencies in data parallelism, recent advances such as Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023b), and ZeRO (Rajbhandari et al., 2020) decompose model parameters, gradients, and optimizer states across devices. These methods significantly reduce the per-device memory footprint and are well-supported by frameworks like DeepSpeed (dee, 2020). While these techniques significantly reduce per-device memory usage, they still involve substantial inter-device communication, particularly during gradient synchronization, which can lead to increased communication overhead and latency.

SWARM Learning SWARM (Ryabinin et al., 2023) learning attempts to address model parallelism and the constraints of individual GPUs by having a number of devices available at each stage of pipeline parallelism while efficiently routing samples through them. This is fundamentally different than the approach described here as it still requires communicating activations across potentially low-bandwidth links, while subnetwork training focuses on reducing all communication to parameters or gradients and maintaining only data parallelism across nodes.

Federated Learning and Dropout-Based Subnetwork Training. Federated learning frameworks (Konečný et al., 2016) train models across decentralized data sources, often focusing on tackling the associated non-iid problem.

Several works have considered training subnetworks per device in a federated setting (Caldas et al., 2018; Horvath et al., 2021; Guliani et al., 2022; Wen et al., 2022; Alam et al., 2022). However, with significantly different goals, context, and methodology. Specifically, these works focus on reducing the communication load and the computational resources of the devices, while in our case, we focus on reducing the necessary memory requirements, an important aspect of allowing training of large models across memory-limited GPUs. Communication load is well addressed by other techniques in the literature, such as multi-step training and compression methods (Reddi et al., 2021; Douillard et al., 2023a; Wang et al., 2023).

FedRolex and HeteroFL considers a scheme where models can vary in their size depending on the client, with the goal of addressing the heterogeneity across devices both in terms of computation and memory. Our work on the other hand focuses on a homogenous setting with the goal to reduce the overall per node memory requirements.

Furthermore, these works are done in a context of data privacy and client heterogeneity, which also lead to problems of small per worker dataset. On the other hand in our setting we assume that each worker can have access to the entire dataset, reducing any issues of heterogeneity or overfitting due to small per worker datasets.

Our method of assigning the max also differs from any of these prior works as we consider assigning masks in a way that assures parameters equal representation in the aggregation, and we systematically compare layer level vs

neuron/channel level subnetwork creation.

Finally the assigned masks in these works are not fixed and change dynamically, in a non-federated setting where wall clock time is critical this would add significant communication and coordination overhead.

3. Method

3.1. Subnetwork Data Parallelism

We consider a distributed training framework designed to enhance memory utilization by assigning each worker a structured *subnetwork*: a subset of model parameters, enabling faster forward and backward passes with reduced memory usage.

Consider a distributed setup comprising N workers (GPUs), with the full model parameter vector denoted as $\boldsymbol{\theta} \in \mathbb{R}^d$. Each parameter θ_j is assigned exactly to $P \leq N$ distinct workers, with possible overlapping assignments such that multiple parameters may share subsets of responsible workers. This assignment is represented by permanent binary masks $\mathbf{m}_i \in \{0,1\}^d$, such that $\sum_{i=1}^N \mathbf{m}_i = P \cdot \mathbf{1}$ for each worker i. This ensures that each parameter is represented precisely P times across the N workers. At optimization step t, worker i samples a mini-batch $\mathcal{B}_i^{(t)}$ and computes gradients based on its masked parameter set:

$$\boldsymbol{\theta}_i^{(t)} = \mathbf{m}_i \odot \boldsymbol{\theta}^{(t)},$$

where $\odot$ denotes element-wise multiplication. The local gradient computation is thus given by:

$$\mathbf{g}_i^{(t)} = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_i^{(t)}; \mathcal{B}_i^{(t)}),$$

with the gradient inherently zero for parameters excluded by the mask, i.e.,

$$(\mathbf{g}_i^{(t)})_j = 0 \quad \text{if} \quad (\mathbf{m}_i)_j = 0.$$

Each worker performs local gradient computation on its assigned subset of parameters, and the gradients are synchronized across their corresponding workers via averaging at the end of each step. This can also be extended to multiple local steps (Konečný et al., 2016; Douillard et al., 2023b), but for this work, we limit our analysis to the single-step case.

After local gradient computations, gradients from all workers are aggregated via masked averaging:

$$\bar{\mathbf{g}}^{(t)} = \frac{\sum_{i=1}^N \mathbf{m}_i \odot \mathbf{g}_i^{(t)}}{\sum_{i=1}^N \mathbf{m}_i} = \frac{1}{P} \sum_{i=1}^N \mathbf{m}_i \odot \mathbf{g}_i^{(t)}.$$

The global parameters are updated using a generic optimization step, denoted by $\text{OptUpdate}(\cdot)$, incorporating internal optimizer state $\mathbf{s}$:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \text{OptUpdate}(\bar{\mathbf{g}}^{(t)}, \mathbf{s}).$$

This framework simplifies to standard data-parallel training when all masks are fully activated (i.e., $P = N$ and $\mathbf{m}_i = \mathbf{1}$ for all i), resulting in:

$$\bar{\mathbf{g}}^{(t)} = \frac{1}{N} \sum_{i=1}^N \mathbf{g}_i^{(t)}.$$

This parameter update rule ensures that each parameter evolves consistently across all participating workers, despite being updated only by a subset. Reducing the value of P proportionally decreases memory and communication costs but introduces a potential optimization gap due to the bias introduced into the gradient estimation.

3.2. Subnetwork construction

We consider two structured approaches for constructing masks based on the classic dropout (Srivastava et al., 2014) that gives ***neuron-level masking***, which can be used with fully connected or convolutional layers, and ***block-level masking*** based on stochastic depth dropout (Huang et al., 2016), which can be used with any residual architectures.

Neuron-Level Masking In neuron-level masking, we disable neurons in fully connected (linear) layers or entire channels in convolutional networks. Consider, for example, two consecutive convolutional layers in a CNN having the same input and output channel size C with weights: $\mathbf{W}^{(l-1)}, \mathbf{W}^{(l)} \in \mathbb{R}^{C \times C \times k \times k}$. We can define a channel level *dropout* masks $\mathbf{M}^{(l)}, \mathbf{M}^{(l-1)} \in \{0,1\}^C$ for output channels of layer l and $l-1$. These masks induce a parameter level mask $\mathbf{m}^{(l)}$ on layer l as follows:

$$\left\{\mathbf{W}^{(l)}_{c,:,:,:},\ b_c^{(l)}\right\} = \begin{cases} 0, & \text{if } (\mathbf{M}^{(l)})_c = 0, \\ \left\{\mathbf{W}^{(l)}_{c,:,:,:},\ b_c^{(l)}\right\}, & \text{otherwise,} \end{cases} \tag{1}$$

$$\mathbf{W}^{(l+1)}_{:,c,:,:} = \begin{cases} 0, & \text{if } (\mathbf{M}^{(l-1)})_c = 0, \\ \mathbf{W}^{(l+1)}_{:,c,:,:}, & \text{otherwise.} \end{cases} \tag{2}$$

This unified masking approach ensures that entire computation paths are deactivated, reducing both forward and backward costs.

Block-Level Masking Block-level masking disables entire computational blocks or layers along paths that have skip connections. Suppose the model contains M blocks $B_1, \ldots, B_K$ with per block parameters $\boldsymbol{\theta}^{(k)}$. Binary masks

$z_i^{(m)} \in \{0, 1\}$ indicate whether layer k is active, which induces a mask on the parameters (removing the layers parameters). In models with skip connections, such as ResNets, masking residual blocks results in identity mappings. Since the input to the subsequent layer is typically a sum of the skip connection and the dropped layer, it allows the representation to still be plausible. Consider a skip connection $B_m(x) + x$, then masked computation at layer m is:

$$\hat{B}_m(\mathbf{x}) = z_i^{(m)} B_m(\mathbf{x}) + \mathbf{x}. \tag{3}$$

The corresponding parameter mask is $\boldsymbol{\theta}_i^{(k)} = z_i^{(k)} \boldsymbol{\theta}^{(k)}$

4. Experiments

We evaluate our proposed partial parameter parallelism framework on multiple datasets: CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), and SVHN (Netzer et al., 2011). We evaluate the effectiveness of subnetwork training under varying parameter sharing levels, defined by the ratio P/N, where each parameter is updated by exactly P out of N workers. Using $N = 8$ GPUs, we vary $P \in \{8, 7, 6, 5, 4, 3\}$ to assess how reduced overlap affects convergence and generalization.

The baseline ($P = N$) corresponds to standard Data Parallel (DP) training, where all parameters are updated by every worker. In contrast, partial overlap settings restrict each parameter to be updated by only a subset of workers: for example, $P/N = 0.875$ (7 of 8 workers), down to $P/N = 0.375$ (3 of 8 workers). We evaluate both neuron-level (channel) and block-level (layer) masking strategies under these settings and compare them to the DP baseline. Subnetwork parallelism offers significant memory savings of $1 - P/N$ for activations, parameters, gradients, and optimizer states, for a given overlap ratio P/N. Thus, lower overlap leads to greater memory.

Setup We conduct all experiments using ResNet-18 (He et al., 2016b) and its wider variant (width = 2) (Zagoruyko & Komodakis, 2016). Models are trained using SGD with Nesterov momentum (Nesterov, 2013) and a momentum factor of 0.9. The baseline ($P/N = 1$) is trained for 200 epochs on $N = 8$ GPUs. All subnetwork configurations are FLOP-matched by increasing the number of training epochs proportionally to the reduction in active parameters. For example, when $P/N = 4/8$, training iterations are doubled to maintain equivalent computational cost.

All models are trained with an effective batch size of $B = 512$ (64 per GPU across $N = 8$ workers). Experiments are conducted with two different types of learning rate schedulers. A cosine annealing learning rate schedule with $\eta_{\text{max}} = 0.2$ and $\eta_{\text{min}} = 0.002$. A linear warm-up is applied over the first 5% of training iterations to improve convergence stability. We also perform experiments with a multi-step learning rate scheduler, from Goyal et al. (2017), where the learning rate is decayed by a factor of 0.1 at specific milestones. For CIFAR-10, these milestones occur at 50% and 75% of the total training iterations. For CIFAR-100, the milestones are set at 30%, 60%, and 80% of the training process. Finally, for SVHN, the milestones are reached at 25%, 33%, 50%, and 75% of the total iterations.

We use group normalization with 2 groups across all experiments, ensuring that normalization is computed only over active parameters in subnetwork configurations. Additionally, we adopt a modified Kaiming initialization (He et al., 2015), recalculating the fan-out based on the number of active (unmasked) output units. This adjustment prevents overestimation of activation variance that can occur with standard initialization when masking is applied.

5. Results

Table 1 reports the top-1 test accuracy across different levels of parameter overlap (P/N) with a ResNet-18 as well as its wide variant (width=2) across three benchmark datasets: CIFAR-10, CIFAR-100, and SVHN. The results compare two subnetwork strategies: **Neuron Masking** and **Block Masking** using a cosine annealing scheduler.

On CIFAR-10, we observe clear benefits of our proposed scheme—*Subnetwork Data Parallelism*. With only 87.5% of the model parameters ($P/N = 0.875$), we achieve a slight performance boost over the baseline across all datasets, highlighting both the regularization effect and computational efficiency of subnetwork training. Notably, even at just 50% of the total parameters $P/N = 4/8$, accuracy remains largely unaffected. These trends are consistent across both variants of ResNet-18 (width $\in \{1, 2\}$) and across all datasets.

For example, in ResNet-18 on CIFAR-10, subnetwork training at $P/N = 0.625$ reduces the number of active parameters per worker from 11.2M to approximately 7.0M, saving over 4M parameters per GPU without impacting performance. This advantage becomes even more pronounced in larger models such as WideResNet-18, which has roughly 44.6M parameters. Under the same setting, each worker processes only 27.9M parameters, yielding a savings of over 16M per GPU. Such substantial reductions in memory and compute overhead highlight the scalability and efficiency of our approach, particularly for larger architectures.

When comparing neuron masking (NM) with block masking (BM), we find that both strategies perform competitively down to $P/N = 0.50$. However, below this threshold, neuron masking suffers a sharp decline in accuracy. As shown in Table 1, at $P/N = 0.375$, for ResNet-18 model (width=1) block (or layer) masking remains robust as compared to neuron masking by consistently giving a superior performance across all the dataets, with only a modest drop—around

Dataset	Masking	DP ($P/N = 1.0$)		$P/N = 0.875$		$P/N = 0.75$		$P/N = 0.625$		$P/N = 0.50$		$P/N = 0.375$	
		RN-18	WRN-18	RN-18	WRN-18	RN-18	WRN-18	RN-18	WRN-18	RN-18	WRN-18	RN-18	WRN-18
CIFAR-10	NM	92.70 ±0.39	92.98 ±0.04	93.68 ±0.19	93.94 ±0.09	93.33 ±0.16	93.95 ±0.31	93.16 ±0.01	93.68 ±0.66	92.64 ±0.13	93.99 ±0.19	81.27 ±3.44	91.80 ±0.62
	BM			92.94 ±0.39	93.45 ±0.13	92.82 ±0.08	93.55 ±0.18	92.19 ±0.24	93.09 ±0.18	91.45 ±0.34	92.54 ±0.22	88.03 ±0.16	88.23 ±0.94
CIFAR-100	NM	68.59 ±0.5	69.74 ±1.56	69.65 ±0.56	69.45 ±0.59	69.76 ±0.22	71.05 ±0.6	70.03 ±0.58	71.06 ±0.21	68.16 ±0.55	70.69 ±0.11	56.21 ±1.41	65.12 ±0.30
	BM			69.89 ±0.24	71.25 ±0.07	69.53 ±1.09	72.01 ±0.09	68.10 ±0.36	70.89 ±0.19	68.12 ±0.41	70.87 ±0.2	61.57 ±0.39	62.92 ±0.8
SVHN	NM	95.06 ±0.04	95.72 ±0.23	95.66 ±0.04	95.79 ±0.18	95.83 ±0.03	95.75 ±0.01	95.64 ±0.04	95.89 ±0.08	95.66 ±0.03	95.86 ±0.04	92.20 ±0.10	95.74 ±0.01
	BM			95.27 ±0.04	95.73 ±0.23	95.06 ±0.05	95.63 ±0.08	95.53 ±0.04	95.75 ±0.06	94.92 ±0.05	94.87 ±0.12	94.10 ±0.08	93.18 ±0.16

Table 1. Top-1 test accuracy (%) results using a cosine annealing scheduler for the learning rate, across different levels of parameter overlap (P/N) for CIFAR-10, CIFAR-100, and SVHN using ResNet-18 (RN-18) and WideResNet-18 (WRN-18). Each row reports accuracy under two masking strategies: neuron-level masking and block-level masking, abbreviated as NM and BM in the table, respectively. $P/N = 1.0$ corresponds to standard data-parallel (DP) training. All experiments are FLOP-matched, with a batch size ($B = 64$ per worker ($N = 8$): $= 512$)

Dataset	Masking	DP ($P/N = 1.0$)		$P/N = 0.875$		$P/N = 0.75$		$P/N = 0.625$		$P/N = 0.50$		$P/N = 0.375$	
		RN-18	WRN-18	RN-18	WRN-18	RN-18	WRN-18	RN-18	WRN-18	RN-18	WRN-18	RN-18	WRN-18
CIFAR-10	NM	92.20 ±0.44	91.30 ±1.03	93.09 ±0.25	93.83 ±0.28	92.73 ±0.19	93.34 ±0.41	91.72 ±0.96	93.26 ±0.45	89.80 ±1.14	92.45 ±0.37	79.83 ±3.15	86.74 ±0.57
	BM			92.02 ±0.49	92.74 ±0.36	92.39 ±0.07	92.59 ±0.32	91.44 ±0.18	91.92 ±0.24	90.47 ±0.42	91.20 ±0.25	87.31 ±0.14	87.96 ±0.54
CIFAR-100	NM	65.39 ±0.95	65.74 ±1.04	66.13 ±0.33	66.72 ±1.32	66.11 ±2.04	66.84 ±0.71	63.85 ±0.91	68.23 ±1.08	59.81 ±1.21	65.91 ±1.55	44.92 ±3.35	52.43 ±5.73
	BM			65.92 ±0.22	67.26 ±0.52	65.52 ±0.51	67.01 ±0.37	64.66 ±0.23	67.11 ±0.48	64.30 ±0.09	67.22 ±0.34	59.93 ±0.26	62.13 ±0.50
SVHN	NM	94.77 ±0.19	93.81 ±0.91	95.22 ±0.43	95.18 ±0.32	95.56 ±0.08	95.38 ±0.17	95.51 ±0.04	95.60 ±0.24	95.08 ±0.30	95.42 ±0.10	94.97 ±0.66	91.98 ±5.08
	BM			94.86 ±0.49	95.05 ±0.19	94.64 ±0.03	94.79 ±0.31	95.21 ±0.03	95.30 ±0.04	94.97 ±0.17	94.79 ±0.14	89.74 ±4.46	88.53 ±5.92

Table 2. Top-1 test accuracy (%) results using a linear scheduler for the learning rate, across different levels of parameter overlap (P/N) for CIFAR-10, CIFAR-100, and SVHN using ResNet-18 (RN-18) and WideResNet-18 (WRN-18). Each row reports accuracy under two masking strategies: neuron-level masking and block-level masking, abbreviated as NM and BM in the table, respectively. $P/N = 1.0$ corresponds to standard data-parallel (DP) training. All experiments are FLOP-matched, with a batch size ($B = 64$ per worker ($N = 8$): $= 512$)

3% on CIFAR-10, 7% on CIFAR-100 and just 1% drop on SVHN. Notably, this drop is even smaller with the wider ResNet-18 (width = 2), indicating that greater representational capacity helps mitigate the impact of block-level masking. This robustness makes block masking particularly well-suited for scenarios with tight memory or compute constraints.

Furthermore, Table 2 presents the results comparing block masking and neuron masking when using a linear multi-step scheduler. We observe consistently superior performance with the block masking strategy, especially at lower overlaps. For example, on CIFAR-100 with ResNet-18 and neuron-level masking at an overlap of $P/N = 0.50$, the accuracy achieved with linear scheduling is 59.81%, whereas block masking yields a **5%** improvement, reaching 64.30%. Additionally, we find that the cosine scheduler delivers even higher performance at the same overlaps for both 1x and 2x model sizes. These observations demonstrate that the effectiveness of the masking techniques is robust across different learning rate schedules and architectures, underscoring their scheduler-agnostic nature.

To quantitatively analyse the two masking strategies, we calculate the gradient alignment between the active parameters of a subnetwork and the corresponding same parameters, when the masking is removed for the same input x. We define gradient alignment as $cos\_sim(g^i_{mask,\theta}, g_{unmask,\theta})$,

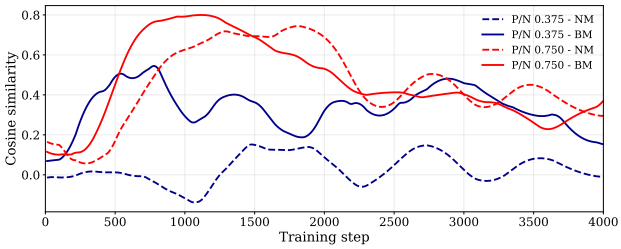

Figure 1. Cosine similarity of gradients in a convolutional layer of a ResNet-18 model, across varying block masking (BM) and neuron masking (NM) at sparsities of $P/N = 0.375$ and $P/N = 0.75$. It can be observed that for a higher overlap, gradient alignment for both subnetwork strategies are similar and positive. For lower overlaps(0.375), it can be seen that gradient alignment is positive for block masking as compared to neuron masking for which alignment is close to zero.

where g_{mask} is the gradient of a worker i and $g_{unmask,\theta}$ is the gradient of a worker with no masking but using the same input as in $g^i_{mask,\theta}$ and cos_sim measures the cosine similarity.

Figure 1 illustrates the gradient alignment for active parameters in the convolutional layer of the second basic block, under two overlap settings: $P/N = 0.75$ and $P/N = 0.375$. At higher overlap ($P/N = 0.75$), both

neuron and block masking exhibit strong alignment early in training—approximately 0.7 and 0.8, respectively, converging with similar trends, though block masking achieves a higher final alignment.

In contrast, at lower overlap ($P/N = 0.375$), a clear divergence emerges: block masking maintains stable, positive alignment throughout training, whereas neuron masking suffers from sharp negative spikes and remains close to zero. These observations, combined with the performance results in Table 1, suggest that block masking is more robust. Unlike neuron masking, which introduces sparsity within layers and can disrupt the internal flow of information, block masking preserves the structural integrity of each layer and relies on identity skip connections to maintain signal propagation. This allows the network to continue learning effectively, even with fewer active blocks.

6. Conclusion and Limitations

We presented a novel distributed training framework based on *subnetwork training*, which improves memory efficiency by isolating structured model components across accelerators and training them in full data parallelism. This approach achieves up to a 40% reduction in memory usage. While our current study focuses on CNNs, future work will extend the method to transformers and language model training. We also plan to explore the impact of mixed masking strategies on gradient alignment.

References

Deepspeed library. https://www.deepspeed.ai, 2020. Accessed: 2025-04-23.

Achiam, O. J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., and et al., S. A. Gpt-4 technical report. 2023. URL https://api.semanticscholar.org/CorpusID:257532815.

Alam, S., Liu, L., Yan, M., and Zhang, M. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *Advances in neural information processing systems*, 35:29677–29690, 2022.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pp. 736–745. PMLR, 2020.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Caldas, S., Konečny, J., McMahan, H. B., and Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

Douillard, A., Feng, Q., Rusu, A. A., Chhaparia, R., Donchev, Y., Kuncoro, A., Ranzato, M., Szlam, A., and Shen, J. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023a.

Douillard, A., Feng, Q., Rusu, A. A., Chhaparia, R., Donchev, Y., Kuncoro, A., Ranzato, M., Szlam, A., and Shen, J. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023b.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Guliani, D., Zhou, L., Ryu, C., Yang, T.-J., Zhang, H., Xiao, Y., Beaufays, F., and Motta, G. Enabling on-device training of speech recognition models with federated dropout. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8757–8761. IEEE, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016b.

Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., and Lane, N. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 646–661. Springer, 2016.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, M. X., Chen, D., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., and Chen, Z. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. Segment anything. *arXiv:2304.02643*, 2023.

Konečný, J. et al. Federated optimization: Distributed machine learning for on-device intelligence. In *arXiv preprint arXiv:1610.02527*, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Li, S. et al. Pytorch distributed: experiences on accelerating data parallel training. *Proceedings of MLSys*, 2020.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H. V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.-Y., Xu, H., Sharma, V., Li, S.-W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. Dinov2: Learning robust visual features without supervision, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LkFG3lB13U5.

Rivaud, S., Fournier, L., Pumir, T., Belilovsky, E., Eickenberg, M., and Oyallon, E. Petra: Parallel end-to-end training with reversible architectures. *International Conference on Learning Representations*, 2024.

Ryabinin, M., Dettmers, T., Diskin, M., and Borzunov, A. Swarm parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, pp. 29416–29440. PMLR, 2023.

Shang, J., Schmeckpeper, K., May, B. B., Minniti, M. V., Kelestemur, T., Watkins, D., and Herlant, L. Theia: Distilling diverse vision foundation models for robot learning. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=ylZHvlwUcI.

Shazeer, N., Cheng, Y., Parmar, N., Tran, D., Vaswani, A., Koanantakool, P., Hawkins, P., Lee, H., Hong, M., Young, C., et al. Mesh-tensorflow: Deep learning for supercomputers. *Advances in neural information processing systems*, 31, 2018.

Shoeybi, M. et al. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. URL https://arxiv.org/abs/2302.13971.

Wang, J., Lu, Y., Yuan, B., Chen, B., Liang, P., De Sa, C., Re, C., and Zhang, C. Cocktailsgd: Fine-tuning foundation models over 500mbps networks. In *International Conference on Machine Learning*, pp. 36058–36076. PMLR, 2023.

Wen, D., Jeon, K.-J., and Huang, K. Federated dropout—a simple approach for enabling federated learning on resource constrained devices. *IEEE wireless communications letters*, 11(5):923–927, 2022.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023a. URL https://arxiv.org/abs/2303.18223.

Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.-C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023b.