`LSH-MoE`: Communication-efficient MoE Training via Locality-Sensitive Hashing

Xiaonan Nie[1] **Qibin Liu**[1] **Fangcheng Fu**[1] **Shenhan Zhu**[1] **Xupeng Miao**[2]
Xiaoyang Li[3] **Yang Zhang**[3] **Shouda Liu**[3] **Bin Cui**[1]
[1]Peking University [2]Purdue University [3]ByteDance
[1]`{xiaonan.nie,2101212782,ccchengff,shenhan.zhu,bin.cui}@pku.edu.cn`
[2]`xupeng@purdue.edu` [3]`{lixiaoyang.x,zhangyang.elfin,liushouda}@bytedance.com`

Abstract

Larger transformer models always perform better on various tasks but require more costs to scale up the model size. To efficiently enlarge models, the mixture-of-experts (MoE) architecture is widely adopted, which consists of a gate network and a series of experts and keep the training cost constant by routing the input data to a fixed number of experts instead of all. In existing large-scale MoE training systems, experts would be distributed among different GPUs for parallelization, and thus input data requires additional all-to-all communications to access the target experts and conduct corresponding computations. However, upon evaluating the training process of three mainstream MoE models on commonly used GPU clusters, we found that the all-to-all communication ratio averaged around 45%, which significantly hinders the efficiency and scalability of training MoE models.

In this paper, we propose `LSH-MoE`, a communication-efficient MoE training framework using locality-sensitive hashing (LSH). We first present the problems of scaling MoE training in existing systems and highlight the potential of exploiting token similarity to facilitate data compression. Then, we introduce an efficient LSH-based compression technique, which utilizes the cross-polytope hashing for rapid clustering and implements a residual-based error compensation scheme to alleviate the adverse impact of compression. To verify the effectiveness of our methods, we conduct experiments on both language models (e.g., RoBERTa, GPT, and T5) and vision models (e.g., Swin) for pre-training and fine-tuning tasks. The results demonstrate that our method substantially outperforms its counterparts across different tasks by $1.28\times$ - $2.2\times$ of speedup.

1 Introduction

In recent years, large-scale pre-trained models have significantly advanced the performance of deep learning across various complex tasks, including computer vision [8, 20], natural language processing [3, 7, 28], and multi-modal learning [19]. Commonly referred to as foundation models, these pre-trained models are primarily built on Transformer architectures [34] and undergo extensive pre-training on large datasets, utilizing substantial GPU resources. OpenAI has validated the scaling law for large language models [15] and suggests that increasing the model's parameter size, the volume of training data, and the duration of training can significantly enhance the model's performance. However, this approach results in a considerable rise in training costs, making the development of foundation models extremely expensive.

Xiaonan Nie, Qibin Liu, Fangcheng Fu, Shenhan Zhu, and Bin Cui are with the School of Computer Science and Key Lab of High Confidence Software Technologies (MOE), Peking University. Bin Cui is also with the Institute of Computational Social Science, Peking University (Qingdao).

To reduce the high computational costs, the sparse mixture-of-experts (MoE) architecture is often adopted, which comprises a sparse gate network and a series of expert networks. This architecture routes input data to only a subset of experts, resulting in sparse activation of the experts and thereby reducing the model's computational FLOPs (float point operations) as well as training costs. Prominent models such as Google's Switch-Transformer [9], ST-MoE [41], Meta's Hash Layer [31] and Mistral-AI's mixtral models [14] have successfully implemented this design, demonstrating improvements in both performance and efficiency with MoE models.

Meanwhile, effectively scaling the training of MoE models across hundreds or even thousands of GPUs remains a significant challenge. Researchers from Google have proposed the *expert parallelism* approach [17], which replicates the gating network on each GPUs and distributes different experts across multiple GPUs for parallel processing. Specifically, each input token is initially processed by the gating network to select the appropriate expert, after which it is routed to the designated experts via peer-to-peer (P2P) network communication. Once the designated experts complete their computation, the token is returned to the original GPU for further processing through an additional P2P communication. Since each GPU typically needs to exchange data with many other GPUs, these P2P transmissions results in an all-to-all communication pattern. Moreover, because the computation of the expert network relies on the outcomes of these communications, the communications cannot be effectively overlapped with ongoing computations. This dependency creates a significant performance bottleneck in model training across most commonly used GPU clusters. We conducted experiments on three widely-used MoE models, including RoBERTa-MoE, GPT-MoE and Swin-MoE, on four A100 servers, each with a cross-machine bandwidth of 200Gb/s. The results, as shown in Figure 3, reveal that the time cost of all-to-all communication constitutes an average of 45% and can reach up to 67% of the total model training time.

Existing methods to improve distributed MoE training on bandwidth-limited clusters tackle communication challenges in various ways. TA-MoE [4] reduces cross-machine communication by adjusting the gating network to favor experts on the same server, while Pre-gated MoE [13] reduces dependency between communication and computation through a pre-gating mechanism that plans token routing in advance. However, both approaches require modifications to the gating mechanism and model structure, limiting their universal applicability. DeepSpeed-MoE [29] introduces PR-MoE, which selects one expert plus a shared expert, halving the all-to-all communication load. SCoMoE [40] organizes all-to-all communication by structuring data transfers along different dimensions and controlling data volumes across network levels, and also clusters tokens to improve routing. However, none of these works consider reducing the All-to-All communication volume in MoE training by compressing the forward activations. Therefore, they can be intergrated with our method for further improvement.

In this paper, we present `LSH-MoE`, a communication-efficient MoE training framework that leverages locality-sensitive hashing to group similar tokens. Our key contributions are as follows:

- We begin by identifying key challenges in scaling MoE training in existing systems, noting that all-to-all communication constitutes an average of 45% of the total training time. Additionally, we investigate the potential of using token similarity to facilitate data compression to reduce communication costs.
- We propose an efficient LSH-based compression technique that employs cross-polytope hashing for rapid clustering. This approach transmits only the clustering centroids, significantly reducing communication costs. To further enhance accuracy, we implement a residual-based error compensation scheme to mitigate the negative effects of compression.
- Through extensive experiments with language models (RoBERTa-MoE, GPT-MoE, and T5-MoE) and vision models (Swin-MoE), across both pre-training and fine-tuning tasks, we demonstrate that our method maintains model quality while achieving a speedup of $1.28\times$ - $2.2\times$ in end-to-end training time.

2 Background

2.1 Mixtures-of-Expert Architecture

To enhance the training efficiency of Transformer models, William et al. (2022) [9] introduced an innovative paradigm, the sparse mixture-of-eexperts (MoE) architecture, illustrated in Figure 1.

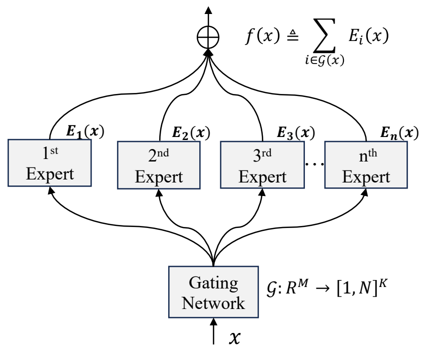


Figure 1: Mixture-of-Experts on a single GPU.

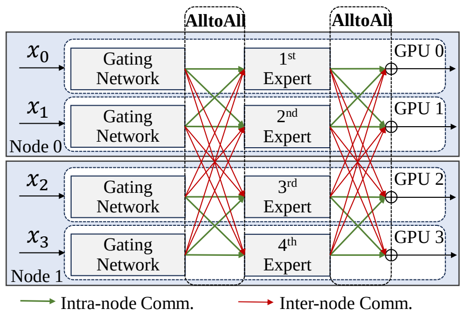


Figure 2: Training Mixture-of-Experts on multiple GPUs as expert parallelism.

This architecture effectively balances parameter capacity and training costs, and comprises two key components: an *expert network* ($\mathcal{E}$) and a *sparse gate network* ($\mathcal{G}$). It is evident that MoE models, with an equal number of active parameters per input, can significantly surpass the performance of dense models. This breakthrough has also catalyzed further research and their application across various industries, as highlighted by numerous subsequent studies [5, 14, 22, 23, 25, 30, 39].

The *expert network* $\mathcal{E}$ is composed of multiple specialized and separate networks, commonly referred to as *experts*, denoted as $\{E_i\}_{i=1}^N$, where N represents the number of experts. Additionally, $E_i(x)$ denotes the output produced when the input x is processed by the i-th expert. Each expert is trained to excel in a specific sub-task, such as in multi-task learning, or to handle specific segments of data, as seen in language modeling and multi-modal learning, thereby increasing the overall model capacity. In foundational models, the MoE layer often serves as a substitute for the traditional feed-forward network (FFN) layer. Within each MoE layer, each FFN function works as an individual expert, significantly enhancing the model's capability to process diverse and complex data inputs.

The *gating network* $\mathcal{G}$ plays a crucial role in the sparse MoE architecture. For example, in a K-way gated MoE system, the gating network outputs a set of integers as Equation 1 to determine which experts should be activated. This decision is based on the characteristics of the input itself, allowing for a dynamic and efficient allocation of computational resources. By only processing each input token with a selected subset of the expert network, the MoE model achieves computation sparsity, effectively decoupling parameter capacity from training costs.

$$\mathcal{G} : \mathbb{R}^M \to [1, N]^K \tag{1}$$

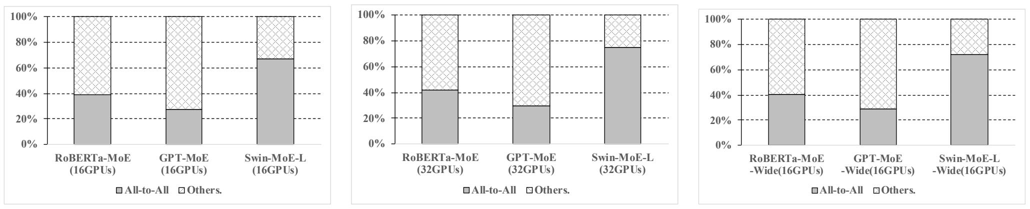
Through the integration of multiple specialized experts, as described by Equation 2, the sparse MoE model is capable of delivering more accurate and efficient predictions as $f(x)$. This is achieved by leveraging the specialized knowledge embedded within each expert, combined with the strategic input allocation managed by the gating network.

$$f(x) \triangleq \sum_{i \in \mathcal{G}(x)} E_i(x) \tag{2}$$

While MoE's primary advantage is decoupling parameter capacity from network cost, a key challenge lies in learning the gating parameters effectively, as the output's sparsity makes it non-differentiable. Consequently, much of the research in the MoE field has centered on developing methods for learning gating functions. These methods fall into three main categories, as outlined in [6]: routing via learnable weighting [9, 24, 30], deterministic hash routing [31], and reinforcement learning-based routing [2, 32, 33]. These approaches primarily differ in the design of the gating network $\mathcal{G}$ rather than the expert network $\mathcal{E}$, and therefore all encounter similar scaling challenges.

2.2 Challenges of Scaling MoE Model Training

While MoE models were initially developed to facilitate efficient scaling during training, deploying these large-scale models in practical GPU-intensive environments poses significant challenges in



(a) 16GPUs (b) 32GPUs (double #GPUs) (c) 16GPUs (double #experts)

Figure 3: Proportion of all-to-all communication time relative to total training duration across different configurations: scaling the number of training servers (Figure 3(b)) and scaling the parameter size of models (Figure 3(c)).

distributed computing. Specifically, the MoE layer harbors a considerably higher number of parameters and requires additional memory, yet it maintains almost the same computational demands as the dense layer. This leads to a unique *compute density* — defined as the ratio of the layer's FLOPs (Floating Point Operations) to its number of parameters. Therefore, traditional parallelism methods such as *tensor parallelism* and *pipeline parallelism* are insufficient for achieving effective parallelism in the scenarios of MoE training.

To improve the efficiency and scalability of training large-scale MoE models, *expert parallelism* [17] has been introduced as a specialized model parallelism strategy. This approach distributes experts within an MoE layer across multiple GPUs, while leveraging data parallelism for replicating non-MoE layers, thus efficiently managing the training workload of MoE models. The workflow of distributed training for an MoE layer is depicted in Figure 2. Once the target expert for each token is determined, an all-to-all communication process is triggered to distribute tokens to their corresponding target experts for computations, denoted as $E_i(x)$. Subsequently, another round of all-to-all communication is executed to gather the outputs from all experts, which produces the MoE layer's output (represented as $f(x)$, Equation 2). Subsequent operations involve executing the data-parallel non-MoE layers.

We first profiled the training process of three popular MoE models employing expert parallelism (detailed in Table 1) on a cluster comprised of four A100 machines, each equipped with an interconnect RDMA bandwidth of 200Gb/s. The proportion of all-to-all communication time relative to the total training duration is illustrated in Figure 3(a). We then double the number of machines, and the number of experts to increase the model scale. The results are shown in Figure 3(b) and 3(c), respectively. Our findings reveal that all-to-all communication accounted for a substantial portion of the total time: approximately 30% in GPT-MoE (15B), 40% in RoBERTa-MoE, and 70% in Swin-MoE-L. And this overhead remains nearly constant in larger models and at larger machine scales. These results highlight a significant bottleneck that hampers the scalability of the training process. Consequently, the duration of all-to-all communication substantially constrains training with expert parallelism, leading to reduced overall throughput and limiting the potential to scale up the number of experts effectively.

2.3 Locality-Sensitive Hashing Algorithms

Locality-Sensitive Hashing (LSH) is a probabilistic method primarily used to approximate nearest neighbor search in high-dimensional spaces, which reduces the dimensionality of data by mapping similar data to the same "buckets" with high probability using hash functions. This approach offers a substantial reduction in computational complexity, particularly beneficial for large-scale data applications. The key operations in LSH including:

Mapping Data into Buckets: The core of LSH is a family of hash functions that maximize the probability of nearby points in the original space staying close in the hashed space, while distant points are likely to end up in different buckets. Each hash function h is characterized by the property: $P[h(x) = h(y)] = 1 - d(x, y)/D$, where $d(x, y)$ is the distance between points x and y, and D denotes the diameter of the space. To map similar data into the same bucket, multiple hash functions from this family are selected based on the specific attributes of the data (e.g., Euclidean distance, cosine similarity) and the desired granularity of the buckets. Data points are then hashed by these

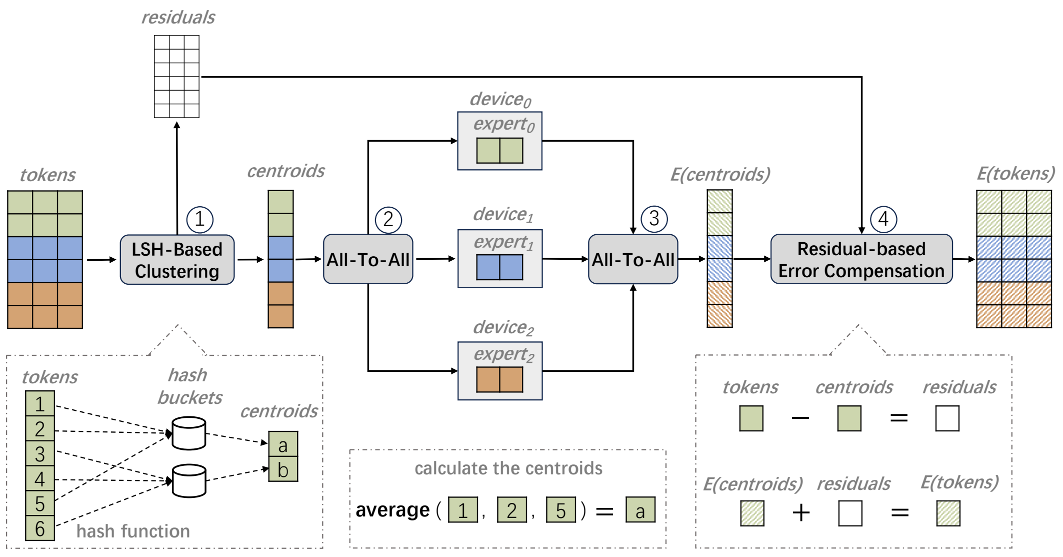


Figure 5: Schematic of MoE training with Locality-Sensitive Hashing (LSH-MoE).

functions, and each point is assigned to buckets according to its hash values, effectively categorizing similar items together for clustering.

Calculating Cluster Centroids: By grouping data points into buckets as determined by their hash values, data points are effectively clustered. Each bucket represents a cluster of data points and the centroid of each cluster is then calculated as the mean of all points within that cluster, formulated as $C_j = \frac{1}{n}\sum_{i=1}^{n_j} x_i$, where C_j is the centroid of the j-th bucket, n_j is the number of points in the j-th bucket, and x_i are the data points in the bucket.

3 Methodology

3.1 The Motivation of Token Similarity

To explore the potential optimization for all-to-all communications in MoE training, we conducted an in-depth analysis of the data involved in these all-to-all communications, identifying a high degree of similarity, termed *token similarity*. Specifically, we applied *Principal Component Analysis* (PCA) to reduce the dimensionality of the input tokens of all-to-all communications and observed a distinct clustering phenomenon, as illustrated in the Figure 4. Our analysis suggests that the observed similarity among tokens may stem from two primary factors:

- *Data Related Influences*: The similarity is partially due to the nature of real-world data, which often adheres to Zipf's Law [18]. This results in a skewed distribution, with certain data elements appear more frequently than others.
- *Model Structure Related Influences*: The design of Transformer architecture [34], especially its attention mechanisms, significantly impacts token similarity. In models like BERT [7], attention layers are designed to capture and integrate context information across tokens, thus homogenizing token representations and emphasizing their shared semantic relationships at the sentence level.

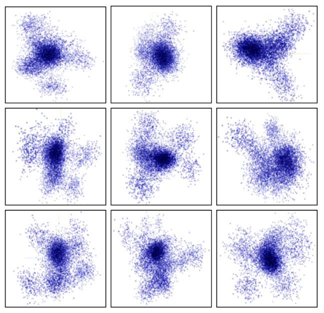


Figure 4: Principal Component Analysis (PCA) Visualization of input tokens involved in all-to-all communication.

3.2 LSH-MoE

Motivated by the *Token Similarity* observed in Section 3.1, we introduce `LSH-MoE`, a novel MoE training framework that integrates locality-sensitive hashing (LSH) for rapid clustering of input tokens. Our method transmits only the clustering centroids, significantly reducing communication volumes. To counteract the negative effects of compression, we also implement a residual-based error compensation scheme.

As depicted in Figure 5, `LSH-MoE` initially employs (1) an LSH-based clustering method to compress *tokens* into *centriods* for subsequent processing, effectively reducing communication overhead. It then sequentially executes (2) all-to-all communication, expert computation, and another (3) all-to-all communication to produce the processed outputs *E(centriods)*. Finally, it introduces (4) a residual-based error compensation method to approximate the expert-processed results *E(tokens)*, by integrating *E(centriods)* with *residuals*. Meanwhile, we also outline the workflow of our `LSH-MoE` framework in the Algorithm 1 of Appendix A.1. The key components of our LSH-MoE framework includes **an efficient LSH-based clustering algorithm** for rapid processing and **an residual-based error compensation scheme** to minimize quality degradation.

Efficient LSH-based Clustering Algorithm. Since the data to be compressed (the input data for all-to-all communication) is generated dynamically and in real time, pre-compressing it or overlapping compression time with other processing tasks is not feasible. Consequently, selecting an efficient online compression algorithm is crucial. Traditional clustering algorithms, such as K-Means, often encounter computational challenges and efficiency limitations. Locality-sensitive hashing (LSH) address these issues by hashing similar data points into the same buckets, enabling faster similarity detection in high-dimensional spaces.

Numerous LSH algorithms have been developed, each employing a unique hashing approach for mapping data onto buckets. We conducted experiments to evaluate several popular hashing algorithms, including *cross-polytope hashing* and *spherical hashing*. Based on our evaluations in Section 4.5, we selected *cross-polytope hashing* as the optimal algorithm for our application. *Cross-polytope hashing* stands out for its method of mapping input vectors to the nearest vertex on a cross-polytope. This process is facilitated by applying randomly rotated cross-polytopes, which effectively segment the surface of the unit sphere. The algorithm can be mathematically represented as follows:

$$LSH(\mathbf{x}) = \text{argmax}_{i \in \{\pm 1, \pm 2, \dots, \pm d\}} |\mathbf{R}\mathbf{x}|_i \tag{3}$$

where $\mathbf{R}$ is a random rotation matrix, d is the dimensionality of the space, and $|\mathbf{R}\mathbf{x}|_i$ denotes the absolute value of the i-th component of the rotated vector $\mathbf{R}\mathbf{x}$.

This formula encapsulates how the input vector x is transformed by the rotation matrix R and then mapped to the nearest vertex of the cross-polytope by selecting the dimension i that maximizes the absolute value of the components of Rx. This method effectively segments the high-dimensional space and enhances the clustering efficiency by rapidly identifying similar data points.

Residual-based Error Compensation Scheme. In our `LSH-MoE` framework, we compress the intermediate activation values within the model network. Unlike gradient compression, this process does not tolerate errors well. Therefore, it is essential to minimize compression-induced errors to ensure minimal impact on model performance. To address this, we implement a novel residual-based gradient compensation strategy, outlined as follows:

1. We first capture the residual for each data point relative to its cluster centroid, defined by the equation:
$$\Delta\text{cluster}_j \leftarrow \{x - \overline{\text{cluster}}_j \mid x \in \text{cluster}_j\}. \tag{4}$$
2. After the expert network computes outputs for the cluster centers, the final step is to restore the processed result for each token by adding back the previously recorded residual:
$$Y_{ij} \leftarrow \{E(\overline{\text{cluster}}_j) + \Delta\text{Cluster}_{jk} \mid k = 1, 2, \dots, N_j\}. \tag{5}$$

This error compensation scheme effectively mitigates potential accuracy loss caused by data compression in all-to-all communication, ensuring the fidelity and robustness of the LSH-MoE framework. The experimental results in Section 4 show that implementing this compensation mechanism enables

Table 1: Models for evaluation, where "-" indicates that the values are different across layers.

Model	**#Layer**	$\mathbf{d_{model}}$	$\mathbf{d_{ffn}}$	**#Experts**	**#Params. (MoE)**	**#Params. (Total)**
RoBERTa-MoE	12	768	3072	16	302M	394M
T5-MoE	16	1024	16384	16	8594M	9288M
GPT-MoE (15B)	12	768	3072	512	14507M	14629M
GPT-MoE (52B)	24	1024	4096	512	51539M	51740M
Swin-MoE-L	24	-	-	32	-	946M

the model trained with `LSH-MoE` to achieve an accuracy comparable to that of a model trained without compression. This outcome highlights the effectiveness of our proposed error compensation strategy in preserving model performance despite the challenges posed by data compression in all-to-all communication.

3.3 Scalability Analysis of LSH-MoE

To effectively demonstrate the scalability of our approach, particularly in terms of its applicability to both larger models and larger computational clusters, we conducted a theoretical analysis. This analysis primarily focuses on the **computation overhead** and the communication costs associated with Mixture of Experts (MoE), specifically considering **all-to-all communication overhead**. We derived the ratio of communication time to computation time, highlighting how this ratio evolves as both the scale of the servers and the model size increase. This relationship is crucial for understanding scalability and can be formally expressed as follows:

$$\frac{T_{all\_to\_all}}{T_{compute}} = \frac{\text{FLOPs}}{6B_{inter}} \times \frac{k}{1+2k} \times \frac{w-1}{wh} \tag{6}$$

where k represents the number of experts activated per token, FLOPs and B_{inter} denote the GPU's computation ability and the network performance, w is the number of GPU servers, and h is the hidden size of model. Notably, the first term, $\frac{\text{FLOPs}}{6B_{inter}}$, remains constant under fixed hardware conditions. Additionally, scaling MoE models typically emphasizes increasing the number of layers and experts, while the growth in hidden size (h) tends to be gradual, as seen in models like Switch-Transformer [9]. Consequently, when both the model scale and the number of training servers grow, the proportion of all-to-all communication time remains nearly unchanged. This insight underpins the scalability of the LSH-MoE method, demonstrating its robustness in larger-scale settings and supporting its potential in future large-scale applications. For a detailed derivation, please refer to Appendix A.2.

4 Experiment

4.1 Implementation

Our LSH-MoE comprises a data compression/restoration component and a communication component. We utilize PyTorch 1.11 for developing the LSH clustering and NCCL for implementing the communication. Additionally, our method is framework-independent and can be easily applied to other MoE training frameworks such as Hetu-MoE [21, 26], DeepSpeed-MoE [29], and Tutel [12].

4.2 Benchmarks and Datasets

Our evaluations are conducted by scaling pre-trained models equipped with MoE architecture across various application domains. This includes models like RoBERTa-MoE, T5-MoE and GPT-MoE in natural language processing (NLP), as well as Swin-MoE in computer vision (CV). Among these models, RoBERTa-MoE and T5-MoE are evaluated on pre-training task, while GPT-MoE and Swin-MoE undergo fine-tuning evaluation based on their official open-sourced model checkpoints [1] [2]. We also evaluated the zero-shot accuracy of the pre-trained T5-MoE. Model configurations are detailed in Table 1.

[1] `https://github.com/facebookresearch/fairseq/tree/main/examples/moe_lm`

[2] `https://github.com/microsoft/Swin-Transformer/blob/main/MODELHUB.md`

The RoBERTa-MoE model is pre-trained with masked language modeling tasks on a combined dataset, which includes BooksCorpus ($\sim$ 800M words) and English Wikipedia ($\sim$ 2,500M words). This dataset is tokenized using a tokenizer with a vocabulary size of 50,257. To assess the impact of our MoE method in compressing all-to-all communication on large model training, the T5-MoE model, which is with about 10B parameters, is pre-trained on an industry dataset ($\sim$ 500M words) using a span-masked language modeling task. In addition to pre-training tasks, we further evaluate our work on fine-tuning tasks. To be specific, we fine-tune two open-sourced models, including the language model GPT-MoE on the General Language Understanding Evaluation (GLUE) benchmark and the vision model Swin-MoE on the ImageNet classification benchmark.

4.3 Software and Hardware Environments

To thoroughly evaluate the effectiveness of our method, we conducted experiments on two clusters, V100 cluster and A100 cluster. Additionally, to ensure consistency in software versions, we performed experiments on both machines using the same docker image.

Software Environment. Our experiments were conducted using a docker image built upon the official NVIDIA GPU containers, which includes Ubuntu 20.04, CUDA 11.3, cuDNN 8.2.0, and NCCL 2.12.7, accessible at NVIDIA GPU Containers [3].

V100 Cluster. The first hardware environment includes two servers, each outfitted with eight NVIDIA V100 (32GB) GPUs. Within each server, GPUs are interconnected using NVLink 2.0 technology. The servers are interconnected via an RDMA NIC, providing a network bandwidth of 100 Gbps.

A100 Cluster. The second hardware environment consists of four servers, each equipped with eight NVIDIA A100 (40GB) GPUs. Within these servers, GPUs utilize NVLink 3.0 technology for interconnection. The servers are linked through two RDMA NICs, enhancing the network bandwidth to 200 Gbps.

We allocated the experiments involving RoBERTa-MoE and GPT-MoE to the V100 cluster, while T5-MoE and Swin-MoE were tested on the A100 cluster. This setup allowed us to effectively compare the performance impacts across different hardware configurations.

4.4 Overall Performance

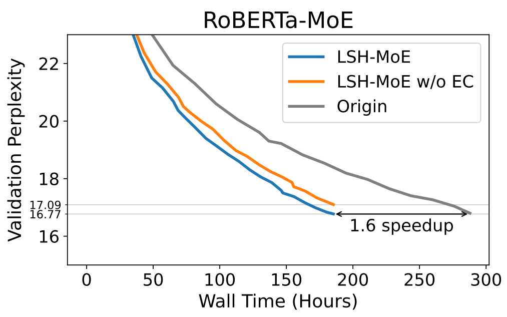
In general, to evaluate our LSH-MoE training approach, which compresses communication data, there are two crucial questions:

1. Does the LSH-MoE method enable normal model convergence, and is there a risk of increased loss variability during this process, potentially leading to instability in training?
2. Might the implementation of the LSH-MoE method adversely affect the model's performance on downstream benchmarks?

Therefore, we conducted experiments focusing on both **Convergence Performance** and **Benchmark Performance** to validate the effectiveness of our method. In this section, due to the necessity of selecting several hyperparameters for LSH, such as the type of hash function and the quantity of hash functions, we have opted for the cross-polytope hash function based on empirical evaluation, setting the number of hash functions at 6. A detailed examination of the effects stemming from variations in these parameters will be methodically addressed in the upcoming ablation study (Section 4.5).

Convergence Performance. We pre-trained the RoBERTa-MoE and T5-MoE using open-source datasets and industrial datasets, respectively. In our approach, we substitute the FFN (Feed-Forward Network) layer with an MoE (Mixture of Experts) layer in alternating layers, as detailed in Section 4.2. We meticulously tracked the time required to achieve equivalent model performance levels (perplexity) during training, as depicted in Figure 6. The results indicate a significant acceleration in training convergence when employing the LSH-MoE method: $1.6\times$ faster for RoBERTa-MoE and $2.2\times$ faster for T5-MoE, compared to the original models' convergence rates. Furthermore, we investigated the role of error compensation in this process. Our findings reveal that omitting error compensation in the LSH-MoE model led to a 0.3 point increase in perplexity, given the same training duration. This observation underscores the efficacy of the error compensation algorithm.

[3]`https://catalog.ngc.nvidia.com/orgs/nvidia/containers/pytorch`



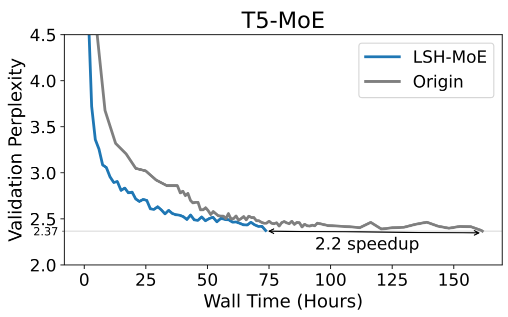


Figure 6: Comparative analysis of convergence performance. This includes a comparison between the original models, LSH-MoE without Error Compensation, and LSH-MoE implementations. The perplexity curves are applied 1D Gaussian smoothing with $\sigma = 0.5$.

Table 2: Evaluation of LSH-MoE on the GLUE benchmark.

Dataset	GPT-MoE (15B)			GPT-MoE (52B)			T5-MoE (10B)	
	Origin	Ours	Speed	Origin	Ours	Speed	Origin	Ours
SST-2	93.8%	93.8%	1.3×	94.5%	94.3%	1.4×	51.6%	50.9%
MNLI	82.8%	82.7%	1.4×	84.1%	84.3%	1.4×	52.6%	52.1%
QNLI	86.6%	86.7%	1.3×	90.2%	90.0%	1.5×	49.5%	50.0%
QQP	88.8%	88.7%	1.3×	88.9%	88.9%	1.2×	-	-
MRPC	71.3%	71.1%	1.3×	76.3%	76.1%	1.3×	-	-
COLA	72.3%	72.4%	1.4×	73.5%	73.8%	1.5×	-	-

Table 3: Results of fine-tuning Swin-MoE on the ImageNet-1K dataset.

	Origin	Ours
Top-1 Acc. ↑	84.7%	84.5%
Top-5 Acc. ↑	97.0%	97.1%
Compression Rate	—	11.7%
Sample/s	184.3	236.6

Benchmark Performance. To better validate the performance of LSH-MoE on downstream tasks, we fine-tuned the GPT-MoE and Swin-MoE on different datasets using open-source model checkpoints, and evaluated zero-shot performance of our internal pre-trained T5-MoE model, adhering to their original architectural designs that incorporate Top-2 gating, as detailed in [1, 12, 17].

We first utilized the LSH-MoE method for fine-tuning the GPT-MoE of two model scales (i.e. 15B and 52B) on the GLUE benchmark, yielding impressive outcomes. As detailed in Table 2, the implementation of the LSH-MoE method substantially reduced communication overhead while maintaining nearly the same level of accuracy. This strategy resulted in a significant performance boost, achieving an acceleration rate ranging from $1.2\times$ to $1.5\times$. The results also demonstrate that as the parameter size of MoE models increases, LSH-MoE continues to achieve significant improvements without compromising model accuracy. Additionally, we report the zero-shot accuracy of the pre-trained T5-MoE, showing that the T5-MoE models trained with LSH-MoE achieved accuracy comparable to standard T5 models, confirming LSH-MoE's efficacy in pretraining. Because the limited number of tokens in the pre-trained dataset and its out-of-domain nature compared to the GLUE evaluation data, the zero-shot performance metrics are relatively low.

Furthermore, our evaluation of the LSH-MoE method in fine-tuning the Swin-MoE on the ImageNet-1K dataset demonstrated noteworthy efficiency. We achieved a communication compression rate of 11.7%, which led to a $1.28\times$ increase in acceleration, as reported in Table 3. Notably, this was accomplished while preserving almost the same level of accuracy.

4.5 Ablation Study

To study the impact of the quantity and types of hash functions, we conducted ablation experiments by fine-tuning the GPT-MoE (15B) model on the MNLI and SST-2 datasets in the GLUE benchmark.

Impact of the Quantity of Hash Functions. We controlled the number of hash functions to indirectly adjust the LSH compression rate, exploring its effect on model performance. Specifically, we utilized 2, 4, 6, 8, and 10 hash functions. As shown in the middle sub-figure in Figure 7, we observe that an increase in the number of hash functions enlarges the number of buckets, enhances data distinction and, consequently, the compression rate. Besides, we indicate from the left sub-figure in Figure 7, that more hash functions leads to improved model convergence quality and worse compression rate.

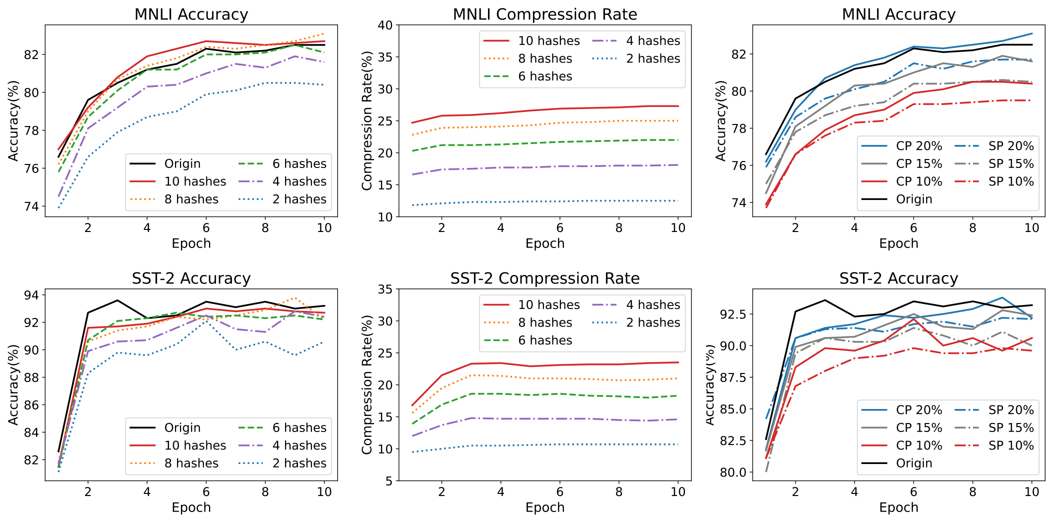


Figure 7: An in-depth analysis of the compression rate and the model performance by adjusting the **quantity** and **types** of hash functions. The left and middle sub-figures are results for diverse quantities of hash functions. The right sub-figure is the result for diverse types of hash functions (CP for cross-polytope and SP for spherical) with different compression rates (20%, 15%, 10%).

Importantly, our results indicate that a compression rate of approximately 20% (achieved with about 6 hash functions) is optimal for maintaining nearly identical convergence as uncompressed models. Therefore, we choose 6 as the default number of hash functions in Section 4.4.

Impact of the Types of Hash Functions. We further explored the impact of the types of hash functions with Cross-Polytope Hashing (CP) and Spherical-Plane Hashing (SP). The outcomes are illustrated in the right sub-figure in Figure 7. CP generally achieves better convergence than SP at the same compression rate. This is attributable to CP's ability to more effectively handle a variety of complex data patterns. CP encodes data based on an n-dimensional cross-polytope, while SP relies on the geometric relationships between spheres and planes. Thus, CP is more generalizable across a variety of complex data patterns while SP performs better with data that has spherical distribution characteristics. Other works (e.g. Reformer [16]) also use CP to leverage the sparsity of attention mechanisms. Therefore, we finally choose **cross-polytope hashing** as the default type of hash functions in Section 4.4.

5 Conclusion

Our study tackled the latency challenges inherent in training sparse-gated Mixture-of-Experts (MoE) models with our innovative LSH-MoE framework. Utilizing locality-sensitive hashing to harness token similarities, our approach significantly reduces communication overhead. The integration of a residual-based error compensation scheme further preserves model integrity under compression. Empirical tests across various models, including RoBERTa, GPT, T5, and Swin, showcase LSH-MoE's capability to accelerate both pre-training and fine-tuning phases by up to 2.2×, paving the way for efficient and scalable MoE applications in real-world settings.

6 Limitations

At the current stage, our work only considers MoE models. Nevertheless, we want to clarify that MoE models are also a mainstream class of deep learning models that are increasingly adopted due to rising model computational demands and training costs, such as Mixtral-7Bx8MoE, DeepSeek-MoE, and GPT-4. Hence, accelerating MoE training is indeed a critical direction. Additionally, the core of our work leverages data redundancy, which is also presented in non-MoE model training. We hope our observations and utilization of data redundancy can inspire more refined work in optimizing training for non-MoE models as well.

Acknowledgments and Disclosure of Funding

This work is supported by Beijing Natural Science Foundation (4244080), National Natural Science Foundation of China (U22B2037, U23B2048), Beijing Municipal Science and Technology Project (Z231100010323002), China National Postdoctoral Program for Innovative Talents (BX20230012), China Postdoctoral Science Foundation (2024M750103), research grant No. SH-2024JK29, ByteDance-PKU joint program (PJ20230202900065), and High-performance Computing Platform of Peking University. Fangcheng Fu and Bin Cui are the corresponding authors.

References

[1] Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*, 2021.

[2] Emmanuel Bengio. *On reinforcement learning for deep neural architectures: conditional computation with stochastic computation policies*. McGill University (Canada), 2017.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Chang Chen, Min Li, Zhihua Wu, Dianhai Yu, and Chao Yang. Ta-moe: Topology-aware large scale mixture-of-expert training. *Advances in Neural Information Processing Systems*, 35:22173–22186, 2022.

[5] Tianlong Chen, Xuxi Chen, Xianzhi Du, Abdullah Rashwan, Fan Yang, Huizhong Chen, Zhangyang Wang, and Yeqing Li. Adamv-moe: Adaptive multi-task vision mixture-of-experts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17346–17357, 2023.

[6] Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, pages 4057–4086. PMLR, 2022.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, 2019.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[9] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[10] Fangcheng Fu, Yuzheng Hu, Yihan He, Jiawei Jiang, Yingxia Shao, Ce Zhang, and Bin Cui. Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript. In *International Conference on Machine Learning*, pages 3304–3314, 2020.

[11] Keshi Ge, Yiming Zhang, Yongquan Fu, Zhiquan Lai, Xiaoge Deng, and Dongsheng Li. Accelerate distributed deep learning with cluster-aware sketch quantization. *Science China Information Sciences*, 66(6):162102, 2023.

[12] Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, et al. Tutel: Adaptive mixture-of-experts at scale. *Proceedings of Machine Learning and Systems*, 5, 2023.

[13] Ranggi Hwang, Jianyu Wei, Shijie Cao, Changho Hwang, Xiaohu Tang, Ting Cao, Mao Yang, and Minsoo Rhu. Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference. *arXiv preprint arXiv:2308.12066*, 2023.

[14] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

[15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[16] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations*, 2020.

[17] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

[18] Wentian Li. Zipf's law everywhere. *Glottometrics*, 5(2002):14–21, 2002.

[19] Junyang Lin, Rui Men, An Yang, Chang Zhou, Ming Ding, Yichang Zhang, Peng Wang, Ang Wang, Le Jiang, Xianyan Jia, et al. M6: A chinese multimodal pretrainer. *arXiv preprint arXiv:2103.00823*, 2021.

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[21] Xupeng Miao, Xiaonan Nie, Hailin Zhang, Tong Zhao, and Bin Cui. Hetu: A highly efficient automatic parallel distributed deep learning system. *Science China Information Sciences*, 66(1):1–2, 2023.

[22] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. *Advances in Neural Information Processing Systems*, 35:9564–9576, 2022.

[23] Xiaonan Nie, Yi Liu, Fangcheng Fu, Jinbao Xue, Dian Jiao, Xupeng Miao, Yangyu Tao, and Bin Cui. Angel-ptm: A scalable and economical large-scale pre-training system in tencent. *arXiv preprint arXiv:2303.02868*, 2023.

[24] Xiaonan Nie, Xupeng Miao, Shijie Cao, Lingxiao Ma, Qibin Liu, Jilong Xue, Youshan Miao, Yi Liu, Zhi Yang, and Bin Cui. Evomoe: An evolutional mixture-of-experts training framework via dense-to-sparse gate. *arXiv preprint arXiv:2112.14397*, 2021.

[25] Xiaonan Nie, Xupeng Miao, Zilong Wang, Zichao Yang, Jilong Xue, Lingxiao Ma, Gang Cao, and Bin Cui. Flexmoe: Scaling large-scale sparse pre-trained model training via dynamic device placement. *Proceedings of the ACM on Management of Data*, 1(1):1–19, 2023.

[26] Xiaonan Nie, Pinxue Zhao, Xupeng Miao, Tong Zhao, and Bin Cui. Hetumoe: An efficient trillion-scale mixture-of-expert distributed training system. *arXiv preprint arXiv:2203.14685*, 2022.

[27] Dmytro Nikolaiev. How to estimate the number of parameters in transformer models. https://towardsdatascience.com/how-to-estimate-the-number-of-parameters-in-transformer-models-ca0f57d8dff0, 2023. Accessed: Oct 22, 2024.

[28] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[29] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *International Conference on Machine Learning*, pages 18332–18346, 2022.

[30] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

[31] Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566, 2021.

[32] Clemens Rosenbaum, Ignacio Cases, Matthew Riemer, and Tim Klinger. Routing networks and the challenges of modular and compositional computation. *arXiv preprint arXiv:1904.12774*, 2019.

[33] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:6000–6010, 2017.

[35] Guozheng Wang, Yongmei Lei, Zeyu Zhang, and Cunlu Peng. A communication efficient admm-based distributed algorithm using two-dimensional torus grouping allreduce. *Data Science and Engineering*, 8(1):61–72, 2023.

[36] Adam Weingram, Yuke Li, Hao Qi, Darren Ng, Liuyao Dai, and Xiaoyi Lu. xccl: A survey of industry-led collective communication libraries for deep learning. *Journal of Computer Science and Technology*, 38(1):166–195, 2023.

[37] Shuo Xiao, Dongqing Zhu, Chaogang Tang, and Zhenzhen Huang. Combining graph contrastive embedding and multi-head cross-attention transfer for cross-domain recommendation. *Data Science and Engineering*, 8(3):247–262, 2023.

[38] Yige Xu, Xipeng Qiu, Ligao Zhou, and Xuanjing Huang. Improving BERT fine-tuning via self-ensemble and self-distillation. *Journal of Computer Science and Technology*, 38(4):853–866, 2023.

[39] An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, et al. M6-t: Exploring sparse expert models and beyond. *arXiv preprint arXiv:2105.15082*, 2021.

[40] Zhiyuan Zeng and Deyi Xiong. Scomoe: Efficient mixtures of experts with structured communication. In *The Eleventh International Conference on Learning Representations*, 2023.

[41] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Algorithm 1 Training MoE models using our LSH-MoE framework

```
Input: $X$: sequence of tokens
Output: $\{Y_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$, where $Y_{ij}$ is the output for tokens in the $j$-th cluster assigned to the $i$-th expert
1: function MoE_Layer_With_LSH($X$)
2:     Calculate the token-to-expert mapping $\zeta$ using the gating network;
3:     Dispatch $X$ into $\{X_i \mid i = 1, 2, \dots, n\}$ based on $\zeta$; // $X_i$ are tokens assigned to the $i$-th expert
4:     for $i \leftarrow 1, 2, \dots, n$ do
5:         $IDX_i \leftarrow LSH(X_i)$; // Get the LSH bucket for each token
6:         Divide $X_i$ into $\{\text{cluster}_j \mid j = 1, 2, \dots, m\}$ based on $IDX$;
7:         for $j \leftarrow 1, 2, \dots, m$ do
8:             $\overline{\text{cluster}_j} \leftarrow \text{Mean}(\text{cluster}_j)$; // Get the centroids for each cluster
9:             $\Delta\text{cluster}_j \leftarrow \{x - \overline{\text{cluster}_j} \mid x \in \text{cluster}_j\}$; // Get the difference between each token and its cluster centroids
10:        $C_i \leftarrow \{\overline{\text{cluster}_j} \mid j = 1, 2, \dots, m\}$;
11:        $\Delta X_i \leftarrow \bigcup_{j=1}^{m} \Delta\text{cluster}_j$;
12:    $C \leftarrow \{C_i \mid i = 1, 2, \dots, n\}$;
13:    $Input \leftarrow \text{all-to-all}(C)$; // Transmit the cluster centroids through all-to-all
14:    $Output \leftarrow \text{Expert}(Input)$; // Perform computations on centroids
15:    $E(C) \leftarrow \text{all-to-all}(Output)$; // Transmit the results back through all-to-all
16:    for $(i, j) \leftarrow (1, 2, \dots, n) \times (1, 2, \dots, m)$ do
17:        $Y_{ij} \leftarrow \{E(\overline{\text{cluster}_j}) + \Delta\text{Cluster}_{jk} \mid k = 1, 2, \dots, N_j\}$; // Apply the residual-based error compensation scheme
18:    return $\{Y\}$.
```

A Appendix

A.1 Framework of LSH-MoE

As illustrated in Algorithm 1, our LSH-MoE training process begins by dispatching each input token in X to its designated expert based on the gating network (Line 2-3). It then utilizes locality-sensitive hashing to cluster tokens into groups, calculating the centroid for each cluster to represent the mean of its tokens, and recording the differences between each token and its centroid for later error compensation (Lines 4-11). These centroids are subsequently transmitted to the experts via all-to-all communication for processing and their results are sent back in a similarly manner (Line 13-15). Finally, a residual-based error compensation is applied to determine the output of the MoE layer (Lines 16-18). This method effectively minimizes the communication load, thus improving the scalability and efficiency of MoE model training.

A.2 Scalability Analysis

First of all, we want to highlight that as the scale of both models and machines increases, the proportion of all-to-all communication time relative to the total time remains nearly constant. This consistency suggests that the LSH-MoE method remains effective even at larger scales. We will now present our derivation step by step, using the notations listed in Table 4.

Formulate all-to-all communication. For any given training server, the amount of tokens (i.e. m) communicated with any other GPU node can be expressed as $m = n \times k/w$. Similarly, the volume of communication within the same GPU node is also equal to $m = n \times k/w$. Consequently, the time required for all-to-all communication during model training can be modeled as follows, with each layer involving two instances for the forward pass and two for the backward pass:

$$T_{all\_to\_all} = 4 \times l \times \left(\frac{m \times h}{B_{intra}} + \frac{m \times h \times (w-1)}{B_{inter}} \right) \approx 4l \times \frac{nk}{w} \times \frac{h(w-1)}{B_{inter}}. \tag{7}$$

Formulate model computation. Based on the derivation in [27], for a standard decoder model, given the number of layers l, and the hidden size h of the model, the activated parameter count per token

Table 4: Notations used in scalability analysis.

Notation	Description
n	The number of tokens processed per GPU
m	The number of tokens communicated between two training servers
k	The number of experts activated per token
h	Hidden size for each token
l	The number of layers for the model
w	The number of training servers
B_{intra}	The intra-machine network bandwidth
B_{inter}	The inter-machine network bandwidth

can be formalized as #ActivatedParams. = $4(1+2k)lh^2$. According to the theory in the appendix of the GPT-3 paper [3], the computation time per GPU can be formalized as $T_{compute}$, where FLOPs represents the computation ability of GPU.

$$T_{compute} = 6 \times \text{\#tokens} \times \frac{\text{\#ActivatedParams.}}{\text{FLOPs}} = \frac{24(1+2k)nlh^2}{\text{FLOPs}}. \tag{8}$$

Formulate all-to-all communication / computation. Therefore, as the machine scale (w) and model scale (l and h) increase, the ratio of computation time to communication time can be formalized as:

$$\frac{T_{all\_to\_all}}{T_{compute}} = \frac{4l \times \frac{nk}{w} \times \frac{h(w-1)}{B_{inter}}}{(24(1+2k)nlh^2)/\text{FLOPs}} = \frac{\text{FLOPs}}{6B_{inter}} \times \frac{k}{1+2k} \times \frac{w-1}{wh}, \tag{9}$$

where the first term $\frac{\text{FLOPs}}{6B_{inter}}$ is constant. As MoE models scale up, the emphasis is generally placed on increasing the number of layers and experts, with a more gradual increase in hidden size (e.g. Switch-Transformer [9]). Consequently, the proportion of communication time remains significant as both the model size and the number of servers increase. These observations and theoretical proofs underscore the sustained effectiveness of the LSH-MoE method in larger environments, thus reinforcing its scalability and applicability for future advancements.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper demonstrates the token similarity in Section 3.1, introduces the critical residual-based error compensation scheme in Section 3.2, and verifies the effectiveness of the method in Section 4.4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer:[Yes]

   Justification: The paper discusses the limitations of this work in Section 6

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides a detailed illustration of the LSH-MoE method in Algorithm 1. Additionally, we thoroughly explain the model setup, dataset configurations, and other relevant details in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Most of the datasets and model checkpoints used in the paper are open-sourced and publicly available. In addition, we provide our code in the supplementary material submitted along with the paper. However, the T5-MoE model is built on a proprietary company platform and training framework. Due to company regulations, we are unable to release the training codes.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The paper provides detailed experimental settings in Section 4.2 for datasets, Section 4.3 for environments, and Table 1 for model hyper-parameters. In addition, we submit our code in the supplementary material along with the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Error bars are not reported because it would be too computationally expensive for pre-training tasks.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides detailed information on the computer resources and environments needed to reproduce the experiments in Section 4.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and preserves anonymity in the submitted code.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: There is no societal impact of the work performed. This paper addresses a foundational research problem and is not tied to particular applications.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper introduces a method to accelerate the training of MoE models, and does not release any new datasets or models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators and original owners of assets used in the paper are properly credited. The paper includes URLs as footnotes for the code base used in the experiments in Section 4.2.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We submit our code in the supplementary material and the documentation is provided alongside the code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.