mrLLM: Fast Multi-Region LLM Inference using Learned Adaptors

Marilyn Rego, Maxwell Kumbong[†], Hermann Kumbong[‡], Ertza Warraich[⋆], Muhammad Shahbaz
University of Michigan [†]*Swarthmore College* [‡]*Stanford University* [⋆]*Purdue University*

1 Motivation and Goals

Large language models (LLMs) are increasingly deployed across geographically distributed GPU clusters to meet rising inference demand. Modern inference systems rely on model-parallel techniques such as pipeline and tensor parallelism, in which different layers are executed across multiple nodes [8, 10]. This setup requires exchanging intermediate activations between devices during the forward pass. When these nodes are distributed across regions (e.g., using services like Amazon Bedrock CRIS [1]), activations must traverse wide-area network (WAN) links. As activation sizes scale with the hidden dimension and sequence length, these transfers are both large and frequent, whereas WAN links provide lower bandwidth and higher latency than intra-data center networks [6]. Consequently, cross-region activation transfer becomes a primary bottleneck, delaying downstream computation and reducing end-to-end inference throughput (Figure 1).

Economic and Performance Challenges over WAN. In addition to performance limitations, cross-region communication introduces substantial monetary costs. Cloud providers charge for inter-region data transfer [2, 3], and repeatedly transmitting large activation tensors can become exorbitant at scale. This issue is particularly pronounced for throughput-oriented workloads (such as batch inference, content moderation, and offline analytics), where large volumes of requests are processed across distributed infrastructure.

Our Insight: Learned Adaptors for Multi-Region Inference. Addressing these challenges requires reducing the size of intermediate activations transmitted over WAN links while preserving the semantics needed by downstream transformer layers. Any solution must remain compatible with existing transformer architectures and inference pipelines, and must retain sufficient structural information to avoid degrading model performance. Unlike conventional compression methods, we leverage learned adapter modules—small, trainable neural network layers (typically low-dimensional bottlenecks) inserted between model partitions [5]—that transform high-dimensional activations into compact representations and reconstruct them at the receiver, preserving task-relevant semantics, thereby enabling more bandwidth-efficient cross-region inference. Motivated by this insight, we develop mrLLM, an adapter-based compression scheme to improve throughput while reducing communication cost over the WAN.

2 Design of mrLLM

We target emerging transformer-based LLMs in which high-dimensional activations need to be communicated across devices at model-parallel boundaries, often forming a dominant latency bottleneck [4, 7]. Our approach, mrLLM, introduces lightweight adapter modules at these communication points to compress activations prior to transmission and reconstruct them afterward. At each boundary, the adapter reduces the outbound activation tensor to a configurable fraction of its original size (typically between 1/8 and 1/200) prior to network transfer, without compromising model accuracy. mrLLM enables a tunable trade-off between throughput and accuracy by varying the bottleneck dimension.

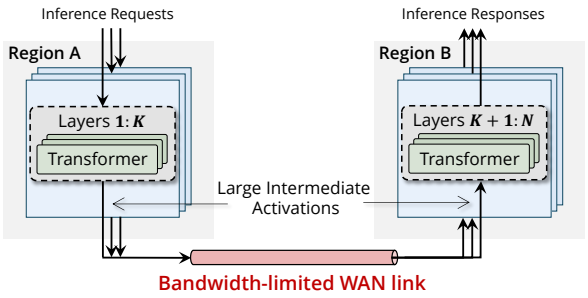

Figure 1: Distributed inference spans multiple regions, requiring large activations to traverse bandwidth-limited WAN links, making communication a key bottleneck.

2.1 Adapter-Based Activation Compression

(a) Adapter Architecture. We employ a two-layer bottleneck MLP that maps hidden activations $x \in \mathbb{R}^H$ to a lower-dimensional space $B \ll H$ and reconstructs them before downstream computation. Given $x \in \mathbb{R}^H$, the adapter computes:

$$y = \text{Dropout}\left(W_{\text{up}} \times \text{GELU}(W_{\text{down}}x + b_{\text{down}}) + b_{\text{up}}\right), \quad (1)$$

where $W_{\text{down}} \in \mathbb{R}^{B \times H}$ projects the hidden representation H into the bottleneck dimension B, and $W_{\text{up}} \in \mathbb{R}^{H \times B}$ reconstructs it back to the original dimension H. Unlike residual adapters [5], our design replaces the activation at the insertion point, forcing all information to pass through the bottleneck. Adapters are injected via forward hooks at selected transformer layers, ensuring compatibility with existing architectures without modifying model weights.

(b) Adapter Training. Adapters are trained with the backbone model frozen, optimizing task performance under a fixed communication budget. We first evaluate the frozen model to establish a baseline, then train adapters using early stopping strategies (e.g., patience-based, beat-base, and target-based) to recover baseline accuracy. The bottleneck dimension B governs both the compression ratio and the information retained, creating a trade-off between communication reduction and

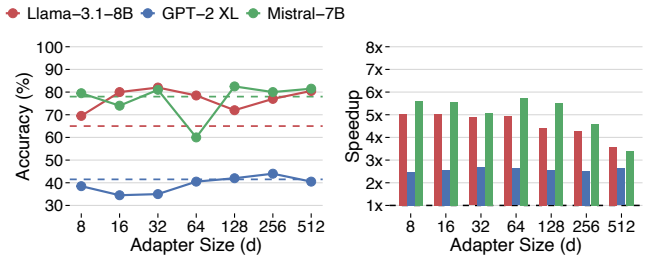

Figure 2: Task accuracy across bottleneck sizes on MNLI.

model performance. We select B via Bayesian Optimization (BO) to jointly maximize speedup and accuracy (Figure 2), and subsequently train adapters at the chosen dimension to recover task performance under the imposed constraint.

2.2 Integrating with Model-Parallel Execution

mrLLM operates with common distributed inference strategies—pipeline parallelism (PP) and tensor parallelism (TP)—which represent the dominant methods for scaling LLM inference and introduce distinct communication patterns. PP partitions the model into sequential stages across devices, while TP shards computation within layers across GPUs, requiring synchronization of intermediate activations.

• **Pipeline Parallelism (PP): Compression at Stage Boundaries.** In pipeline-parallel inference, the transformer is partitioned across devices, and activations are exchanged at stage boundaries. We align adapter placement with these boundaries, compressing activations before transmission and decompressing them upon receipt. This directly reduces cross-device communication and improves end-to-end throughput.

• **Tensor Parallelism (TP): Sharded Compression for Synchronization.** In tensor-parallelism, computation within layers is distributed across GPUs, requiring synchronization of partial activations. By sharding adapters, mrLLM allows each device to operate on a portion of the compressed representation. Adapters are inserted at communication-critical layers (e.g., attention outputs and MLP projections), reducing synchronization overhead while preserving correctness.

3 Preliminary Evaluation

We evaluate the effectiveness of activation compression adapters in a real-world distributed inference setup using pipeline parallelism. Our environment comprises two GPU pods deployed across geographically distinct regions on RunPod.io to enable cross-region inference over WAN links. The transformer model is partitioned across the two nodes, with intermediate activations transmitted between stages during inference.

We evaluate three representative models—Llama-3.1-8B-Instruct, GPT2-xl (1.5B), and Mistral-7B-Instruct-v0.3—on the MNLI benchmark [9] using the SetFit/mnli dataset. Adapters are trained with bottleneck dimensions $d \in 8, 16, 32, 64, 128, 256, 512$. The hidden size is 4,096 for Llama and Mistral, and 1,600 for GPT2. These configurations correspond to compression ratios ranging from $512\times$ to $8\times$ for Llama and Mistral, and from $200\times$ to $3\times$ for GPT2.

We measure inference throughput (requests/s) and accuracy relative to the baseline model without compression. During training, we employ a beat-base early stopping criterion, terminating once validation performance matches or exceeds the uncompressed baseline.

Summary of Results. Across all models, activation compression improves throughput by reducing cross-device communication, yielding an average speedup of $4.08 \pm 1.23\times$. Llama achieves $4.58 \pm 0.54\times$ speedup with consistent accuracy gains ($+12.1 \pm 4.6$ points), peaking at +17 points ($d = 32$). GPT2 shows stable $2.58 \pm 0.07\times$ improvements, with modest accuracy changes on average (-2.1 ± 3.5 points), recovering and slightly exceeding baseline at larger bottlenecks (+2.5 at $d = 256$). Mistral attains the highest average speedup ($5.07 \pm 0.84\times$), but exhibits higher variance in accuracy (-1.1 ± 8.0 points), improving up to +4.5 points ($d = 128$) while degrading under aggressive compression ($d = 64$). Overall, adapter-based compression significantly accelerates pipeline-parallel inference while largely preserving—and often improving—accuracy.

References

[1] Amazon Bedrock CRIS. https://docs.aws.amazon.com/bedrock/latest/userguide/cross-region-inference.html. Last Accessed: 02/2026.

[2] AWS Direct Connect. https://docs.aws.amazon.com/directconnect/. Last Accessed: 02/2026.

[3] Azure ExpressRoute. https://azure.microsoft.com/en-us/products/expressroute. Last Accessed: 02/2026.

[4] Kartikeya Bhardwaj, Ching-Yi Lin, Anderson Sartor, and Radu Marculescu. Memory- and Communication-Aware Model Compression for Distributed Deep Learning Inference on IoT. *ACM TECS*, 2019.

[5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-Efficient Transfer Learning for NLP. In *ICML*, 2019.

[6] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: Experience with a Globally-Deployed Software-Defined WAN. *ACM SIGCOMM*, 2013.

[7] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. In *ACM SC*, 2021.

[8] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. In *ACM/IEEE ISCA*, 2024.

[9] Adina Williams, Nikita Nangia, and Samuel Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *ACL NAACL*, 2018.

[10] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *USENIX OSDI*, 2022.