Demo: CollabTrans: Device-cloud Collaborative Inference Framework for Transformer-based Models

Jingcan Chen, Huatao Xu, Mo Li
jchenhv@cse.ust.hk,huatao@ust.hk,lim@cse.ust.hk
The Hong Kong University of Science and Technology
Hong Kong, SAR, China

Abstract

Model splitting and offloading part of the DNN model from the mobile device to the cloud server, known as collaborative inference, improves end-to-end latency and server throughput in CNN-based models. However, current collaborative approaches do not apply to popular Transformer-based models, as these models generate large intermediate outputs with significant transmission latency and have a uniform block structure that makes it challenging to serve tail models efficiently on the server. We propose CollabTrans, a collaborative inference framework designed for Transformer-based models to enhance server scalability when handling requests from numerous end devices, taming the large output with truncated SVD and serving heterogeneous tail models by sharing a complete model. We demonstrate our framework with a real-time image classification mobile application together with background inference request traffic, showing the high inference throughput of our framework.

CCS Concepts

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Computing methodologies** → **Cooperation and coordination**; **Factorization methods**.

Keywords

Transformer Models, Device-cloud Collaborative Inference, Inference Serving System, Mobile Computing

ACM Reference Format:
Jingcan Chen, Huatao Xu, Mo Li. 2025. Demo: CollabTrans: Device-cloud Collaborative Inference Framework for Transformer-based Models. In *The 23rd Annual International Conference on Mobile Systems, Applications and Services (MobiSys '25), June 23–27, 2025, Anaheim, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3711875.3734371

1 Introduction

Deep neural networks (DNNs) are widely used in mobile applications. Common cloud-based inference raises concerns about privacy and long data transmission latency, while on-device deployment struggles to meet the latency demands of complex DNNs. *Collaborative inference* [1, 3] addresses this by splitting the model into two parts, head model and tail model, deployed separately on the device and the server. Such splitting takes advantage of low latency of transferring intermediate output and exploits the resource on the devices.

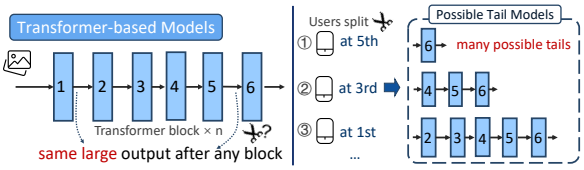

Figure 1: Model Splitting in Transformer Models

On CNN-based models like AlexNet and Incepction, model splitting algorithms leverage the computation graph to split at a model layer to optimize latency or throughput. However, it is challenging to apply to the popular Transformer-based models. In this demo, we focus on discriminative models, e.g. Vision Transformers [2], rather than auto-regressive models like generative language models. We identify two challenges (Figure 1): (1) **Intermediate outputs between Transformer blocks are large.** Unlike CNN models with several layers of small intermediate output, the output between Transformer blocks is relatively large compared to raw data input and splitting at any layer results in 1-3× transmission latency, thus infeasible to apply graph-based splitting methods. (2) **Transformer blocks have identical structure.** Depending on the devices' computation resource, splitting can occur after any block within the Transformer model due to its uniform structure. This results in multiple potential tail models, which complicates model serving and management on the server with limited resource. We present our framework CollabTrans to address the challenges and boost server throughput to scale to numerous users.

2 CollabTrans Designs

Figure 2 depicts CollabTrans's workflow, including the following steps. ❶ Before inference, the clients on mobile devices select a splitting configuration (split point and intermediate data compression ratio) using **truncated singular value decomposition (SVD)**, a lossy model splitting algorithm. This configuration should satisfy the user accuracy loss constraint and meet the latency deadline. The selection is based on the model latency profile and current network bandwidth. ❷ Upon receiving requests (intermediate data), the server buffers requests for different tail models in separate queues. The scheduler determines the batch sizes for each request queue using our **scheduling strategy**. ❸ Each GPU worker has **a shared model** and batches the requests during the runtime of the iteration. After inference, the responses are returned to clients.

Model Splitting via Truncated SVD. Transformer models have large output after each Transformer block, and current splitting methods are either infeasible or entail extra costly overheads. We employ truncated SVD [5] to decompose a linear layer in Transformer block into low-rank matrices at the expense of accuracy,

MobiSys '25, June 23–27, 2025, Anaheim, CA, USA

ACM ISBN 979-8-4007-1453-5/25/06
https://doi.org/10.1145/3711875.3734371

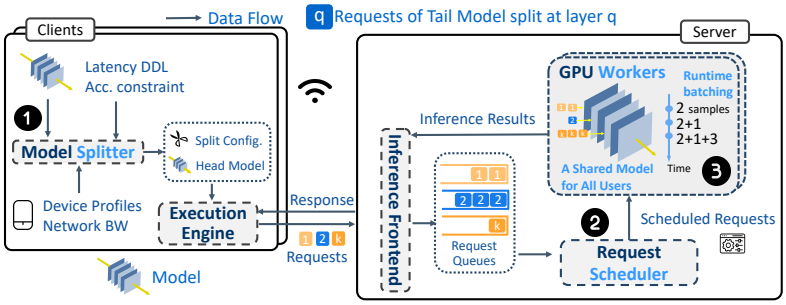

Figure 2: CollabTrans Overview.

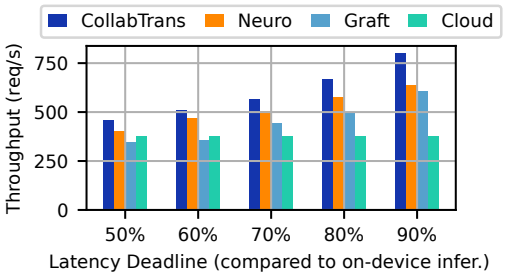

Figure 3: Server throughput under different latency deadlines using various collaboration algorithms.

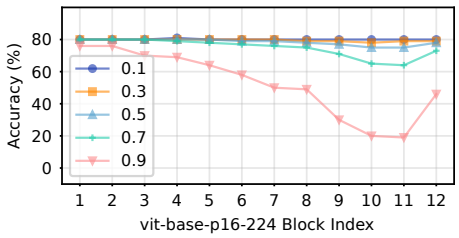

Figure 4: Accuracy loss with various truncation ratios (0.1-0.9) on different model blocks of vit-base-patch16-224.

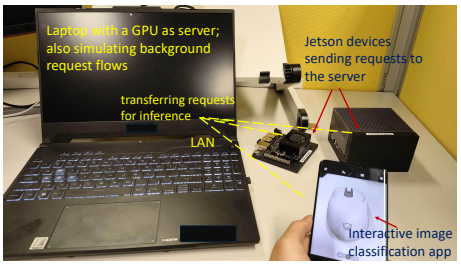

Figure 5: Device setting used for demonstration.

creating a possible splitting point, where the output size is dependent on the rank of the matrix. This method provides a flexible trade-off between data transfer latency and inference accuracy by choosing the rank. Based on device capabilities and network bandwidth and given latency and accuracy constraints, the model splitter chooses the optimal splitting point that keeps as many layers on devices as possible to alleviate the server computation.

Model Sharing with Runtime Batching in GPU Workers. The heterogeneous tail models are *partially shared in the rear blocks* (e.g. the 6th block in Figure 1). We thus use a *shared model* to serve requests of all tail models. This shared model batches the requests from queues of different tail models by checking the queues before the calculation of each block, which avoids unnecessary memory consumption and model-switching overheads.

Scheduling Algorithm to Optimize Throughput. To efficiently allocate GPU resources to different queues based on current queue lengths, we dynamically determine batch sizes for request queues. A budget batch size is set for a specific GPU worker. A prioritized queue can have a major large batch size, while other queues share the remaining budget based on queue lengths. The priority is fairly round-robin among queues. With this algorithm for the shared model, our serving system sustains high inference throughput with low serving latency among all queues.

3 Evaluation and Demonstration

We implement CollabTrans and conduct extensive experiments using 5 Jetsons devices with different power configurations on ViT models, and compare the maximal throughput with baselines [4, 6]. Results (Figure 3) show that, CollabTrans achieves up to 1.32× higher server throughput than Graft, the SOTA serving system for model splitting [6] and 2.11× higher than server-only inference, under the deadline of 90% of on-device inference latency. The throughput gain results from the exploitation of devices as they conduct part of the model inference. In the experiments above, at least 90% of the requests are served within the deadline. Figure 4 shows the accuracy loss when applying truncated SVD on different blocks of the model. For the first half of the blocks (1-7), applying a truncation ratio of 0.7 (compress 70% of the intermediate data) only incurs less than 5% accuracy degradation.

For demonstration, we use a mobile phone with a real-time image classification application for user interaction and Jetsons running the same task (Figure 5). A laptop is used as a server. We also simulate the inference requests from numerous devices by emitting background request flows on the laptop. The laptop visualizes the real-time throughput and request latency, and shows the pre-measured throughput trace of server-only inference. Users can see the model split point, classification result and inference latency on the mobile phone. Users can adjust accuracy and latency constraints, network bandwidth to see the throughput change.

Acknowledgments

This paper is supported in part by the Start-up Fund of HKUST under grant R9899, and in part by Hong Kong GRF under grant 16204224.

References

[1] Amin Banitalebi-Dehkordi et al. 2021. Auto-split: A general framework of collaborative edge-cloud AI. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2543–2553.
[2] Alexey Dosovitskiy et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR* (2021).
[3] Chuang Hu et al. 2019. Dynamic adaptive DNN surgery for inference acceleration on the edge. In *IEEE INFOCOM 2019*. IEEE, 1423–1431.
[4] Yiping Kang et al. 2017. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News* 45, 1 (2017), 615–629.
[5] Xin Wang et al. 2024. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint* (2024).
[6] Jing Wu et al. 2023. Graft: Efficient inference serving for hybrid deep learning with SLO guarantees via DNN re-alignment. *IEEE Transactions on Parallel and Distributed Systems* (2023).