2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW) | 979-8-3503-0208-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCGridW59191.2023.00084

Towards Efficient Scheduling of Concurrent DNN Training and Inferencing on Accelerated Edges

Prashanthi S K †, Vinayaka Hegde and Yogesh Simmhan
Department of Computational and Data Sciences, Indian Institute of Science, Bangalore 560012 INDIA
Email: prashanthis@iisc.ac.in, vinayakah@iisc.ac.in, simmhan@cds.iisc.ac.in

***Abstract*—Edge devices are typically used to perform low-latency DNN inferencing close to the data source. However, with accelerated edge devices and privacy-oriented paradigms like Federated Learning, we can increasingly use them for DNN training too. This can require both training and inference workloads to be run concurrently on an edge device, without compromising on the inference latency. Here, we explore such concurrent scheduling on edge devices, and provide initial results demonstrating the interaction of training and inferencing on latency and throughput.**

I. Introduction and Motivation

Deep Neural Network (DNN) models are often used for video analytics in smart city [1] and autonomous vehicle [2] applications. With the growth in Internet of Things (IoT), edge devices deployed close to the data source are being used to perform *DNN inferencing tasks*. This shift in computation to the edge is driven by both the bandwidth for transmitting videos to the cloud, as well as for privacy and legal concerns.

Accelerated edge devices such as Nvidia Jetson are growing more powerful over time and are competitive for DNN training [3]. E.g., Nvidia's Jetson AGX Orin has 12 ARM Cortex A78AE CPU cores, an Ampere GPU with 2048 CUDA cores and 64 tensor cores, and 64 GB of RAM shared between CPU and GPU. It delivers 275 TOPS of performance [4], comparable to an RTX 3060 Ti GPU workstation.

Two classes of applications necessitate *DNN training* on edge devices. The first is federated learning [5], a privacy-focused distributed training technique on local data stored on the edge devices, with only model weights sent to the server for aggregation. The second is continual/lifelong learning [6], where a model is periodically retrained on the edge over new representative samples from real-world conditions to prevent data drift [7]. In such scenarios, the training workloads are not exclusively using the edge devices, and are instead interleaved with latency-sensitive inference workload requests. Here, inference workloads may run continuously, interspersed with occasional training workloads that need to be run, or vice versa. In some cases, both may be running continuously.

When both workloads arrive concurrently for execution, running them sequentially, one after the other, is a simple strategy that prevents interference between them. But it causes higher wait times for one of them. Inferencing is typically latency sensitive and cannot afford to wait till the training epochs are complete. Also, if inferencing requests arrive continuously, there may not be any "gap" to schedule the training. If the training workload is made to wait too long, it can translate to a lower inference accuracy due to data drift [7]. Further, serial execution under-utilizes the hardware despite enabling intra-job pipelining [8]. Thus, concurrent execution of these workloads is necessary to improve response time and utilization. We also observe that the baseload is a key component of energy use on edge accelerators [9], and concurrent execution can reduce this as well. In this paper, we motivate the need for concurrent execution and provide initial results on the interaction between concurrent training and inference workloads at various batch sizes.

† Student author

II. Related work and Gaps

Gandiva [10] is a cluster scheduler that uses intra-job predictability to time slice GPUs efficiently across multiple DNN training jobs, as well as migrate jobs dynamically to improve cluster efficiency. Antman [11] is another scheduler that uses spare resources to execute multiple jobs on a shared GPU while minimizing interference between them. MURI [8] packs Deep Learning training jobs on multiple resource types in order to achieve high utilization and also reduce job completion time. However, all of these works consider server/cloud GPUs which support GPU sharing among multiple workloads, which is not the case on Jetson edge accelerators. Additionally, none of these works considers inferencing workloads which are much more lightweight compared to training as they only involve the forward pass. While throughput is the primary performance metric for training, latency is a key metric for inference workloads.

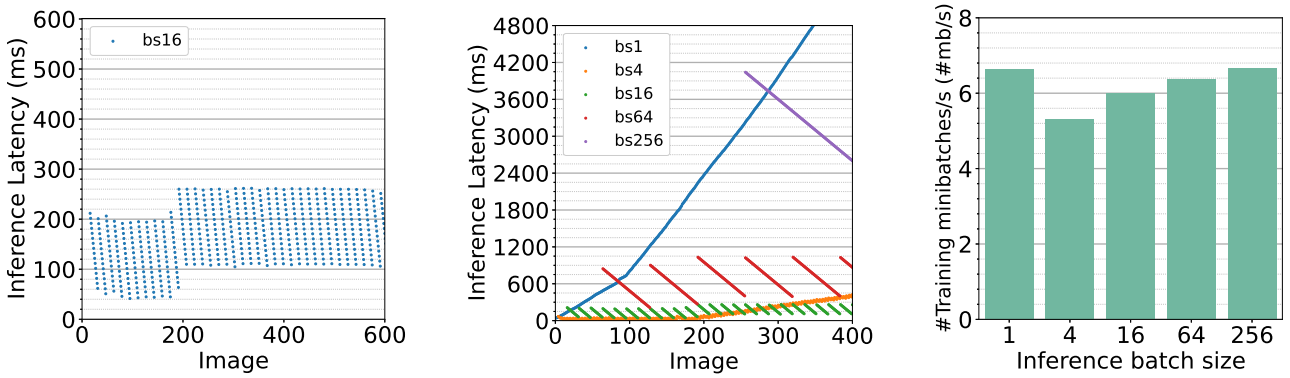
Ekya [7] proposes a scheduler for continual learning (joint inference and retraining) on edge servers. They use Nvidia's Multi-Process Service (MPS) [12] to manage the GPU reallocation between training and inferencing processes. However, Jetson edge accelerators do not support MPS.

Research into scheduling concurrent training and inferencing workloads on edge devices is limited. Additionally, edge devices have unique features such as a shared RAM between the CPU and GPU, and support for multiple power profiles with different power/performance trade-offs and energy budgets, none of which are explored by existing work.

III. Proposed Approach

We propose to interleave training and inferencing workloads intelligently to ensure complementary resource utilization and

979-8-3503-0208-0/23/$31.00 ©2023 IEEE
DOI 10.1109/CCGridW59191.2023.00084



(a) Inference latency per image, with inference batch size of 16.

(b) Inference latency with varying inference batch sizes (bs).

(c) Effect of inf batch size on train throughput

Figure 1: Effect of batch size on training and inference throughput and latency.

avoid contention. We plan to include a wide variety of models/datasets as our training and inferencing workloads. We also plan to design tuning techniques of PyTorch parameters such as inference batch size and the number of Dataloader workers considering the interaction between the concurrent workloads while also meeting the workload's goals such as inference latency. Finally, we plan to look at the energy aspects of such scheduling and come up with a variant that optimizes for energy rather than execution time. Particularly, we propose to change the power mode of the device (which includes CPU, GPU and memory frequencies, and the number of online CPU cores) depending on the optimization goal, workload phase, and resource needs.

IV. Preliminary Results

Setup. We run experiments on an Nvidia Jetson AGX Orin developer kit [13] in MAXN power mode with DVFS off and fan at max speed. Datasets are stored on the SSD, and the page cache is flushed at the start of every experiment. We use the `jtop` utility to measure CPU, GPU utilization, power, etc. PyTorch v1.12 is used as the deep learning framework, and we add instrumentation to measure the compute, fetch and end-to-end times.

Workload. We consider MobileNetv3 with the GLD23k dataset as our training workload and ResNet50 model with the Imagenet validation dataset as our inferencing workload. The inference images arrive at a rate of $100/s$, i.e., with a gap of $10\ ms$. Both these workloads run concurrently for $10\ mins$.

1) The inference batch size has to be chosen keeping in mind the inference latency constraints: In Fig 1a, we report the inference latency per image as the sum of the execution time and the queuing time, for an inferencing batch size of $bs = 16$, for the first $6\ s$. The figure shows a sawtooth pattern of latency (Y axis) for each image index (X axis). The slope of the sawtooth is due to the different queuing times for streaming images being accumulated in a batch – the earliest image with the most queuing time is at the top of the sawtooth while the bottom of the sawtooth indicates the execution time for the full batch. We see lower inference latencies of around $200\ ms$ for the first 200 images as this corresponds to the first $\approx 2\ s$ of the run where the concurrent training has not started and the inference does face any interference. After the 200^{th} image, we see a steady-state behavior with a peak latency within $260\ ms$. So the rest of the run is not plotted.

Next, we repeat this experiment with varying inference batch sizes. As seen from Fig. 1b, $bs = 1$ and $bs = 4$ are unstable since the batch execution time is higher than the accumulation time for images in a batch. This results in an unbounded increase in latency over time due to increasing queuing times. $bs = 16$, 64 and 256 are stable and have a bounded latency. However, as the batch size increases, the execution time also increases, and this leads to higher overall latency with maximum latencies at $0.26\ s$, $1.2s$ and $4.1s$ for batch sizes of 16, 64 and 256, respectively.

2) Increasing the inference batch size also increases the training throughput: Next, we vary the inference batch size from 1 to 256 and report the rate at which the concurrent mini-batch training happens. As seen from Fig 1c, the training throughput goes up from 6 batches per second at batch size 16 to 6.38 batches per second at batch size 64 and further to 6.65 batches per second at batch size 256. This is because as the inference batch size increases, the amount of idle time spent waiting for the images to accumulate for the batch also increases, and hence the training job faces lower interference and makes better progress. Batch size 1 is an exception, but batch sizes 1 and 4 are not considered as they are unstable configurations.

V. Conclusion

In this paper, we motivate the need for concurrent DNN training and inferencing on accelerated edge devices. Further, we demonstrate empirically the interaction and interference between the two workloads under various batch sizes. In future, we plan to develop efficient scheduling strategies for concurrent training and inferencing on accelerated edge devices.

REFERENCES

[1] Q. Chen, W. Wang, F. Wu, S. De, R. Wang, B. Zhang, and X. Huang, "A survey on an emerging area: Deep learning for smart city data," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 5, 2019.

[2] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, 2020.

[3] Prashanthi S.K, A. Khochare, S. A. Kesanapalli, R. Bhope, and Y. Simmhan, "Don't miss the train: A case for systems research into training on the edge," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2022.

[4] NVIDIA, "Jetson orin modules and devkit," https://www.nvidia.com/en-in/autonomous-machines/embedded-systems/jetson-orin/, 2022.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017.

[6] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[7] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, N. Karianakis, Y. Shu, K. Hsieh, V. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *USENIX NSDI*, 2022.

[8] Y. Zhao, Y. Liu, Y. Peng, Y. Zhu, X. Liu, and X. Jin, "Multi-resource interleaving for deep learning training," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022.

[9] Prashanthi S.K, S. A. Kesanapalli, and Y. Simmhan, "Characterizing the performance of accelerated jetson edge devices for training deep learning models," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 3, December 2022.

[10] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang, F. Yang, and L. Zhou, "Gandiva: Introspective cluster scheduling for deep learning," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, 2018.

[11] W. Xiao, S. Ren, Y. Li, Y. Zhang, P. Hou, Z. Li, Y. Feng, W. Lin, and Y. Jia, "Antman: Dynamic scaling on gpu clusters for deep learning," in *Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation*, 2020.

[12] NVIDIA, "Multi process service," https://docs.nvidia.com/pdf/CUDA_Multi_Process_Service_Overview.pdf, 2022.

[13] Nvidia., "Jetson agx orin developer kit," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/, 2022.